Model Compiler Suite for Aries

Developers Guide

version 0.8.4

May 20, 2024



Important Notice

MOBILINT, Inc. reserves the right to make changes to the information in this publication at any time without prior notice. All information provided is for reference purpose only. MOBILINT assumes no responsibility for possible errors or omissions, or for any consequences resulting from the use of the information contained herein.

This publication on its own does not convey any license, either express or implied, relating to any MOBILINT and/or third-party products, under the intellectual property rights of MOBILINT and/or any third parties.

MOBILINT makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does MOBILINT assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

Customers are responsible for their own products and applications. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by the customer's technical experts.

MOBILINT products are not designed, intended, or authorized for use in applications intended to support or sustain life, or for any other application in which the failure of the MOBILINT product could reasonably be expected to create a situation where personal injury or death may occur. Customers acknowledge and agree that they are solely responsible to meet all other legal and regulatory requirements regarding their applications using MOBILINT products notwithstanding any information provided in this publication. Customer shall indemnify and hold MOBILINT and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim (including but not limited to personal injury or death) that may be associated with such unintended, unauthorized and/or illegal use.

**WARNING** No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electric or mechanical, by photocopying, recording, or otherwise, without the prior written consent of MOBILINT. This publication is intended for use by designated recipients only. This publication contains confidential information (including trade secrets) of MOBILINT protected by Competition Law, Trade Secrets Protection Act and other related laws, and therefore may not be, in part or in whole, directly or indirectly publicized, distributed, photocopied or used (including in a posting on the Internet where unspecified access is possible) by any unauthorized third party. MOBILINT reserves its right to take any and all measures both in equity and law available to it and claim full damages against any party that misappropriates MOBILINT’s trade secrets and/or confidential information.



Mobilint, Inc.

3F, Narakium Yeoksam B Building, 35, Seolleung-ro 93-gil, Gangnam-gu, Seoul, Republic of Korea (06151)

Contact Number

+82) 2-6281-0489

Corporate URL

www.mobilint.com

E-mail Address

contact@mobilint.com

Document Revision History

| Doc Revision Number | Date | Description |
| --- | --- | --- |
| 0.8.4 | May 20, 2024 | Revised for v0.8.4 |
| 0.8.3 | March 7, 2024 | Revised for v0.8.3 |
| 0.8.2 | February 23, 2024 | Revised for v0.8.2 |
| 0.8.1 | December 8, 2023 | Revised for v0.8.1 |
| 0.8.0 | November 2, 2023 | Revised for v0.8.0 |
| 0.7.12 | September 12, 2023 | Revised for v0.7.12 |
| 0.7.11 | August 31, 2023 | Revised for v0.7.11 |
| 0.7.10 | August 11, 2023 | Revised for v0.7.10 |
| 0.7.9 | August 11, 2023 | Revised for v0.7.9 |
| 0.7.8 | August 8, 2023 | Revised for v0.7.8 |
| 0.7.7 | June 30, 2023 | Revised for v0.7.7 |
| 0.7 | March 23, 2023 | Revised for v0.7 |
| 0.6 | August 10, 2022 | Revised for v0.6 |
| 0.5 | July 1, 2022 | Revised for v0.5 |
| 0.4 | February 23, 2022 | Revised for v0.4 |
| 0.3 | February 5, 2022 | Revised for v0.3 |
| 0.2 | December 1, 2021 | Revised for v0.2 |

Table of Contents

[1. Introduction 8](#_Toc167110226)

[2. Changelog 9](#_Toc167110227)

[2.1 qubee v0.8.4 (May 2024) 9](#_Toc167110228)

[2.2 qubee v0.8.3 (March 2024) 9](#_Toc167110229)

[2.3 qubee v0.8.2 (February 2024) 9](#_Toc167110230)

[2.4 qubee v0.8.1 (December 2023) 9](#_Toc167110231)

[2.5 qubee v0.8.0 (November 2023) 9](#_Toc167110232)

[2.6 qubee v0.7.12 (September 2023) 9](#_Toc167110233)

[2.7 qubee v0.7.11 (August 2023) 9](#_Toc167110234)

[2.8 qubee v0.7.10 (August 2023) 9](#_Toc167110235)

[2.9 qubee v0.7.9 (August 2023) 9](#_Toc167110236)

[2.10 qubee v0.7.8 (August 2023) 9](#_Toc167110237)

[2.11 qubee v0.7.7 (June 2023) 9](#_Toc167110238)

[2.12 qubee v0.7 (March 2023) 10](#_Toc167110239)

[2.13 qubee v0.6 (August 2022) 10](#_Toc167110240)

[2.14 qubee v0.5 (July 2022) 10](#_Toc167110241)

[2.15 qubee v0.4 (February 2022) 10](#_Toc167110242)

[2.16 qubee v0.3 (February 2022) 10](#_Toc167110243)

[2.17 qubee v0.2 (December 2021) 10](#_Toc167110244)

[3. Installation 11](#_Toc167110245)

[3.1 System requirements 11](#_Toc167110246)

[3.1.1 Reference System 11](#_Toc167110247)

[3.1.2 Requirements and Recommended Packages 11](#_Toc167110248)

[3.2 SDK Installation 11](#_Toc167110249)

[3.2.1 Building Docker Container 11](#_Toc167110250)

[3.2.2 Installation of qubee 12](#_Toc167110251)

[4. Tutorials 13](#_Toc167110252)

[4.1 Preparing Calibration Data 13](#_Toc167110253)

[4.1.1 Pre-process raw calibration dataset and save it as numpy tensors 13](#_Toc167110254)

[4.1.2 Use a pre-processing configuration YAML file 14](#_Toc167110255)

[4.1.3 Use a manually defined pre-processing function 14](#_Toc167110256)

[4.1.4 Use Mobilint® Calibration GUI Tool 15](#_Toc167110257)

[4.2 Compiling ONNX Models 16](#_Toc167110258)

[4.3 Compiling PyTorch Models 16](#_Toc167110259)

[4.4 Compiling TensorFlow/Keras Models 17](#_Toc167110260)

[4.5 Compiling TensorFlow Lite Models 18](#_Toc167110261)

[4.6 Compling Models with Custom Input(ONNX/Torch/TensorFlow Frameworks) 19](#_Toc167110262)

[5. CPU Offloading (Beta Version) 20](#_Toc167110263)

[6. Supported Frameworks 21](#_Toc167110264)

[6.1 Supported Operations (ONNX) 21](#_Toc167110265)

[6.2 Supported operations (PyTorch) 23](#_Toc167110266)

[6.3 Supported operations (TensorFlow/Keras/TensorFlow Lite) 25](#_Toc167110267)

[7. API Reference 28](#_Toc167110268)

[7.1 Function: mxq\_compile 28](#_Toc167110269)

[7.1.1 Tips for choosing quantization methods 30](#_Toc167110270)

[7.2 Class: Model\_Dict 30](#_Toc167110271)

[7.2.1 Methods 31](#_Toc167110272)

[7.2.2 Method Details 31](#_Toc167110273)

[7.3 Function: make\_calib 35](#_Toc167110274)

[7.4 Fuction: make\_calib\_man 35](#_Toc167110275)

[7.5 Pre-processing Configurations 36](#_Toc167110276)

[7.5.1 Pre-processing Parameters 37](#_Toc167110277)

[8. Open Source License Notice 39](#_Toc167110278)

[9. Copyright 40](#_Toc167110279)

List of Tables

Table Title Page

[Table 6‑1. ONNX Supported Operations 21](#_Toc167110280)

[Table 6‑2. PyTorch Supported Operations 23](#_Toc167110281)

[Table 6‑3. TensorFlow Supported Operations 25](#_Toc167110282)

[Table 7‑1. mxq\_compile 28](#_Toc167110283)

[Table 7‑2. Model\_Dict Class 31](#_Toc167110284)

[Table 7‑3. Model\_Dict Methods 31](#_Toc167110285)

[Table 7‑4. Model\_Dict.\_\_init\_\_ 31](#_Toc167110286)

[Table 7‑5. Model\_Dict.compile 32](#_Toc167110287)

[Table 7‑6. Model\_Dict.inference 34](#_Toc167110288)

[Table 7‑7. Model\_Dict.inference\_int8 34](#_Toc167110289)

[Table 7‑8. Model\_Dict.inference\_int8\_input\_dict 34](#_Toc167110290)

[Table 7‑9. Model\_Dict.to 34](#_Toc167110291)

[Table 7‑10. make\_calib 35](#_Toc167110292)

[Table 7‑11. make\_calib\_man 35](#_Toc167110293)

[Table 7‑12. Pre-processing function API 36](#_Toc167110294)

[Table 7‑13. GetImage 37](#_Toc167110295)

[Table 7‑14. Pad 37](#_Toc167110296)

[Table 7‑15. Normalize 37](#_Toc167110297)

[Table 7‑16. ResizeTorch 38](#_Toc167110298)

[Table 7‑17. Resize 38](#_Toc167110299)

[Table 7‑18. CenterCrop 38](#_Toc167110300)

[Table 7‑19. SetOrder 38](#_Toc167110301)

List of Figures

Figure Title Page

[Figure 1‑1. SDK Components 8](#_Toc167110302)

[Figure 1‑2. Input and output of qubee 8](#_Toc167110303)

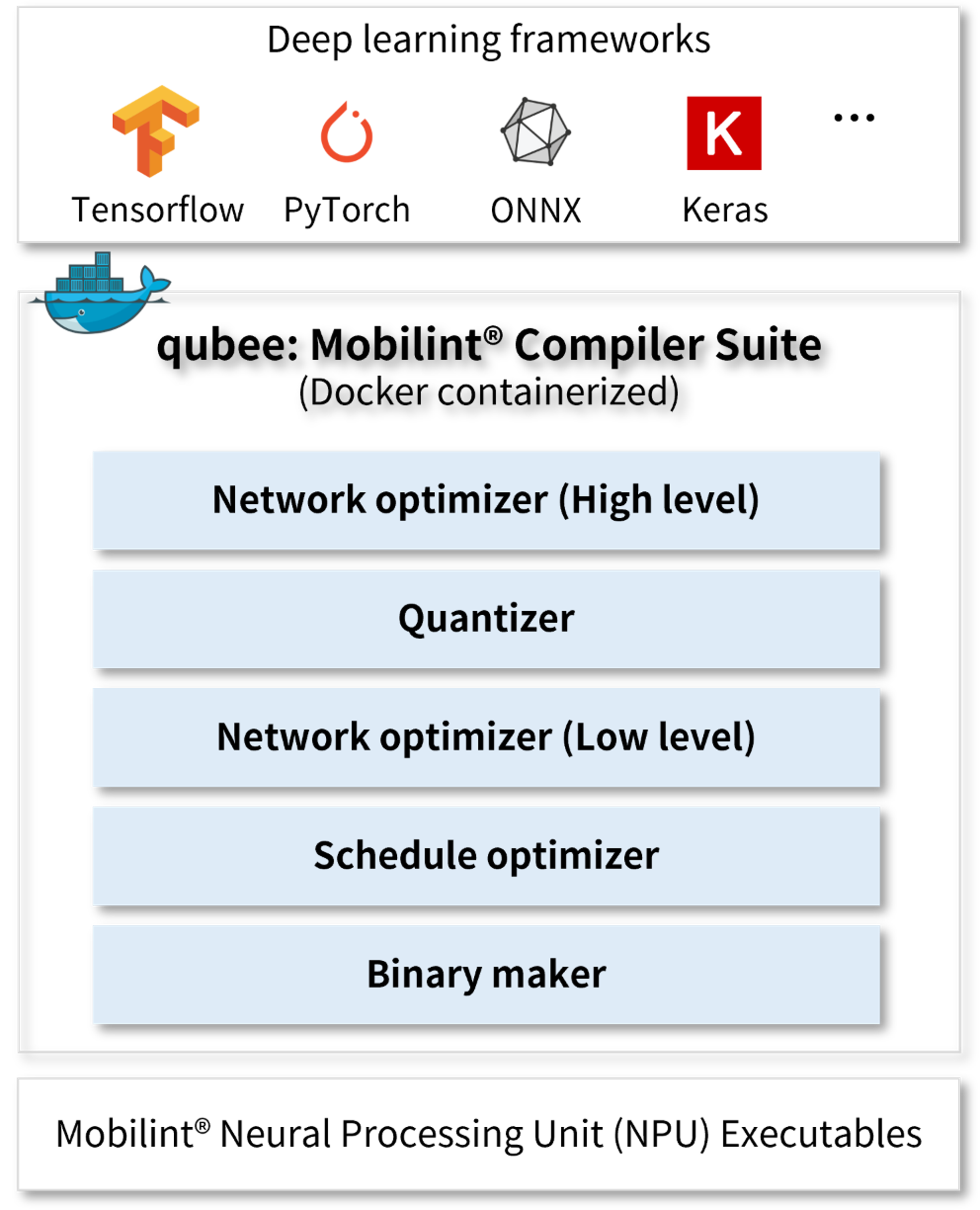
[Figure 4‑1. Mobilint® Calibration GUI 16](#_Toc167110304)

[Figure 5‑1. SDK CPU Offloading 20](#_Toc167110305)

[Figure 6‑1. Supported deep-learning frameworks 21](#_Toc167110306)

# Introduction

Mobilint® Model Compiler (i.e., Compiler) is a tool that converts models from deep learning frameworks (ONNX, PyTorch, Keras, TensorFlow, etc...) into Mobilint® Model eXeCUtable (i.e., MXQ), a format executable by Mobilint® Neural Processing Unit (NPU). This is the manual for the **qubee**, Mobilint's SDK. In this manual, you can learn how to use it, what frameworks it supports, etc. A set of functions you can use to interact with the SDK will be given below.



1. Figure 1‑1. SDK Components

Inputs to qubee are a trained deep learning model, its input shape, and calibration data. It will return MXQ (compiled model) as an output.



1. Figure 1‑2. Input and output of qubee

# Changelog

## qubee v0.8.4 (May 2024)

API

TF backend connected to ONNX backend by TF2ONNX

Support more operations

## qubee v0.8.3 (March 2024)

API

Support TF Lite backend

## qubee v0.8.2 (February 2024)

## qubee v0.8.1 (December 2023)

## qubee v0.8.0 (November 2023)

API

TVM backend deprecated

## qubee v0.7.12 (September 2023)

## qubee v0.7.11 (August 2023)

API

Support TorchScript backend

## qubee v0.7.10 (August 2023)

## qubee v0.7.9 (August 2023)

## qubee v0.7.8 (August 2023)

## qubee v0.7.7 (June 2023)

API

Improve CPU offloading (beta version)

Improve CPU efficiency

Support more operations

Docker

torch: 1.10.1 -> 1.13.0

tensorflow: 2.3.0 -> 2.9.0

onnx:1.11.0 -> 1.12.0

## qubee v0.7 (March 2023)

Multi-channel quantization

Support more operations

API

Improve calibration dataset processing

Support CPU offloading (beta version)

## qubee v0.6 (August 2022)

Minor updates

## qubee v0.5 (July 2022)

Docker

Conda -> Virtualenv

Python: 3.7.7 -> 3.8.10

torch: 1.8.1 -> 1.10.1

tensorflow: 1.15.0 -> 2.3.0

onnx:1.6.0 -> 1.11.0

Parser

Code refactoring

API

Enable saving sample inference results (inputs and outputs)

## qubee v0.4 (February 2022)

Optimizer

Minor updates in fusing reshape

## qubee v0.3 (February 2022)

Parser

Identify preprocess and postprocess of the model

Exclude preprocess and postprocess if they are unsupported by the NPU

API

Simulate integer inference in Python API

## qubee v0.2 (December 2021)

First release

# Installation

## System requirements

We recommend to use NVIDIA GPU for faster compile wtih qubee, but it is not necessary. Currently, CPU version qubee is also supported.

### Reference System

|  |
| --- |
| Ubuntu 22.04.4 LTS  NVIDIA Graphics Driver 545.29.06 |

### Requirements and Recommended Packages

|  |
| --- |
| Ubuntu 20.04.6 LTS or Above  NVIDIA Graphics Driver 450.80.02 **or** Above  Docker  nvidia-docker |

## SDK Installation

We recommend installing qubee on the Mobilint docker container.

(Docker image: mobilint/qbcompiler:v0.8, <https://hub.docker.com/r/mobilint/qbcompiler>)

### Building Docker Container

Run the following commands to build the docker container.

|  |
| --- |
| $ *# Download Docker Image*  $ docker pull mobilint/qbcompiler:v0.8  $ *# mkdir {WORKING DIRCTORY} (if needed)*  $ cd {WORKING DIRCTORY}  $ docker run -it --gpus all --ipc=host --name {YOUR\_CONTAINER\_NAME} -v $(pwd):/workspace mobilint/qbcompiler:v0.8 /bin/bash |

(Recommended) If the trained models and datasets are stored in different directories, you can mount them to the docker container as follows:

|  |
| --- |
| $ docker run -it --gpus all --ipc=host --name {YOUR\_CONTAINER\_NAME} -v $(pwd):/workspace -v {PATH TO MODEL DIR}:/models -v {PATH TO DATASET DIR}:/datasets mobilint/qbcompiler:v0.8 /bin/bash |

(Optional) Build the docker image for CPU only version

|  |
| --- |
| $ *# Download Docker Image*  $ docker pull mobilint/qbcompiler:v0.8-cpu  $ cd {WORKING DIRCTORY}  $ docker run -it --ipc=host --name {YOUR\_CONTAINER\_NAME} -v $(pwd):/workspace mobilint/qbcompiler:v0.8-cpu /bin/bash |

(Optional, the latest version is not available yet) Build the docker image for WSL2

|  |
| --- |
| $ *# Download Docker Image*  $ docker pull mobilint/qbcompiler:v0.7-wsl  $ *# Make a docker container*  $ cd {WORKING DIRCTORY}  $ docker run -it --gpus all --ipc=host --name {YOUR\_CONTAINER\_NAME} -v $(pwd):/data mobilint/qbcompiler:v0.7-wsl |

### Installation of qubee

qubee compiler packages are available in [Mobilint® Software Development Kit (SDK)](https://dl.mobilint.com/view.php%20).

Run the following commands to install qubee on the docker container.

|  |
| --- |
| $ *# Download qubee-0.8.3-py3-none-any.whl file*  $ *# Copy qubee whl file to Docker*  $ docker cp {Path to qubee-0.8.3-py3-none-any.whl} {YOUR\_CONTAINER\_NAME}:/  $ *# Start Docker*  $ docker start {YOUR\_CONTAINER\_NAME}  $ *# Attach to Docker container*  $ docker **exec** -it {YOUR\_CONTAINER\_NAME} /bin/bash  $ *# Install qubee compiler*  $ cd /  $ python -m pip install qubee-0.8.3-py3-none-any.whl |

(Option, for WSL2) Run the following commands to install qubee on the docker container.

|  |
| --- |
| $ *# Download qubee-0.8.1\_wsl-py3-none-any.whl file*  $ *# Copy qubee whl file to Docker*  $ docker cp {Path to qubee-0.8.1\_wsl-py3-none-any.whl} {YOUR\_CONTAINER\_NAME}:/  $ *# Start Docker*  $ docker start {YOUR\_CONTAINER\_NAME}  $ *# Attach to Docker container*  $ docker **exec** -it {YOUR\_CONTAINER\_NAME} /bin/bash  $ *# Install qubee*  $ cd /  $ python -m pip install qubee-0.8.1\_wsl-py3-none-any.whl |

# Tutorials

The tutorials below go through the steps for preparing the calibration dataset, model compiling, and inference.

## Preparing Calibration Data

To compile the model, you should prepare the calibration dataset (the pre-processed inputs for the model) for quantization. There are three ways to make the calibration dataset as follows:

(i) Pre-process the raw calibration dataset and save it as numpy tensors.

(ii) Utilize a pre-processing configuration YAML file (only for images with **uniform format**).

(iii) Use a manually defined pre-processing function (only for images with **uniform format**).

(iv) Use [Mobilint® Calibration GUI Tool](https://git.mobilint.com/algorithm-team/utility/calibration_gui)

**Important**  The process of making a calibration dataset may vary depending on whether you compile the model for CPU offloading or not. Currently, qubee compiles the model without CPU offloading by default. In this scenario, the pre-processed input shape should be in the format (H, W, C). On the other hand, when CPU offloading is employed, the pre-processed input shape should match the input shape that the original model takes.

### Pre-process raw calibration dataset and save it as numpy tensors

You can save the pre-processed calibration dataset as numpy tensors with your custom pre-processing function and use them to compile the model.

An example code is shown below. The following code assumes that we have an image folder consisting of 1000 randomly selected JPEG image files from the [ImageNet](https://www.image-net.org/) dataset for calibration prepared in directory `/datasets/imagenet/cali\_1000`.

|  |
| --- |
| **import** os  **import** numpy as np  **import** cv2  **def** **get\_img\_paths\_from\_dir**(dir\_path: str, img\_ext = ["jpg", "jpeg", "png"]):  **assert** os.path.**exists**(dir\_path)  candidates = os.**listdir**(dir\_path)  **return** [os.path.**join**(dir\_path, y) **for** y **in** candidates **if** **any**([y.**lower**().**endswith**(e) **for** e **in**  img\_ext])]  **def** **pre\_process**(img\_path: str, target\_h: int, target\_w: int):  img = cv2.**imread**(img\_path, cv2.IMREAD\_COLOR)  resized\_img = cv2.**resize**(img, dsize=(target\_w, target\_h)).**astype**(np.float32)  **return** resized\_img  **if** \_\_name\_\_ == "\_\_main\_\_":  img\_dir = "/datasets/imagenet/cali\_1000"  save\_dir = "/workspace/calibration/custom\_single\_input"  target\_h, target\_w = 224, 224  os.**makedirs**(save\_dir, exist\_ok=True)  img\_paths = **get\_img\_paths\_from\_dir**(img\_dir)  **for** i, img\_path **in** **enumerate**(img\_paths):  fname = f"{i}".**zfill**(3) + ".npy"  fpath = os.path.**join**(save\_dir, fname)  x = **pre\_process**(img\_path, target\_h, target\_w)  np.**save**(fpath, x) |

The above results are in a directory containing the pre-processed calibration dataset (numpy tensors of shape (224,224, 3)), located at `/workspace/calibration/custom\_single\_input`.

### Use a pre-processing configuration YAML file

Image pre-processing techniques such as resizing, cropping, and normalization are often applied in machine vision tasks. Users can construct a pre-processing configuration using a YAML file and prepare the calibration dataset via the API provided by qubee, *make\_calib*. Please be aware that this method can only be employed when the raw data is an image. An example code is shown below. The following code assumes that images for calibration are prepared in the directory `/workspace/cali\_1000`.

|  |
| --- |
| **from** qubee **import** make\_calib  **make\_calib**(  args\_pre="/workspace/mobilenet\_v2.yaml", *# path to pre-processing configuration yaml file*  data\_dir="/datasets/imagenet/cali\_1000", *# path to folder of original calibration data files such as images*  save\_dir="/workspace/calibration/", *# path to folder to save pre-proceessed calibration data files*  save\_name="mobilenet\_v2", *# tag for the generated calibration dataset*  max\_size=50 *# Maximum number of data to use for calibration*  ) |
| *# mobilenet\_v2.yaml*  Datatype: Image  GetImage:  to\_float32: **false**  channel\_order: RGB  Pre-Order: [ResizeTorch, CenterCrop, Normalize, SetOrder]  Pre-processing:  ResizeTorch:  size: [256, 256]  interpolation: bilinear  CenterCrop:  size: [224, 224]  Normalize:  mean: [0.485, 0.456, 0.406]  std: [0.229, 0.224, 0.225]  to\_float\_div255: **true**  SetOrder:  shape: HWC |

The above results are in a directory containing the pre-processed calibration dataset (numpy tensors), located at `/workspace/calibration/mobilenet\_v2`. In addition, a calibration meta txt file containing the paths to the pre-processed numpy files is created, named `/workspace/calibration/mobilenet\_v2.txt`.

**Remark** The sample dataset for calibration should be composed of images with the same format. If some are in color images and others are in grayscale images, the calibration dataset will not be created properly.

### Use a manually defined pre-processing function

You can use your pre-processing function to make the calibration dataset via the API provided by qubee, *make\_calib\_man*. In this case, the pre-processing function should take the image path as input and return a numpy tensor. An example of the code is shown below. The following code assumes that images for calibration are prepared in the directory `/datasets/imagenet/cali\_1000`.

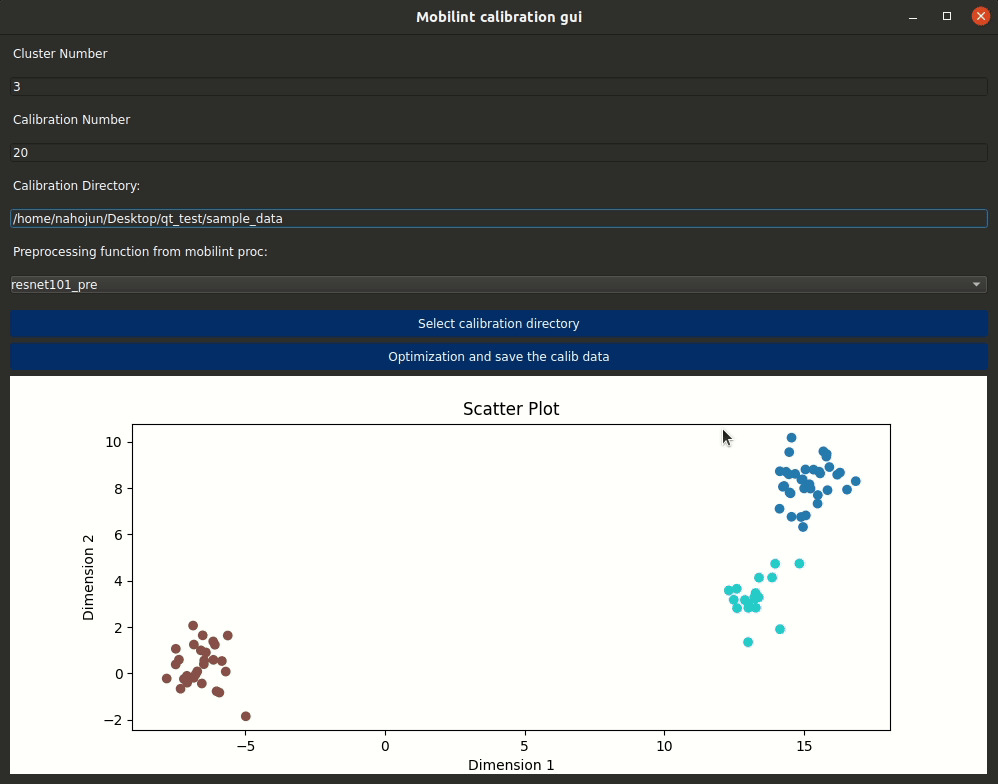
|  |
| --- |
| **import** torch  **import** numpy as np  **from** PIL **import** Image  **import** torchvision.transforms.functional as F  **from** qubee **import** make\_calib\_man  **def** **preprocess\_resnet50**(img\_path: str):  img = Image.**open**(img\_path)  resize\_size=(232, 232)  crop\_size=(224, 224)  mean=[0.485, 0.456, 0.406]  std=[0.229, 0.224, 0.225]  out = F.**pil\_to\_tensor**(img)  out = F.**resize**(out, size=resize\_size)  out = F.**center\_crop**(out, output\_size=crop\_size)  out = out.**to**(torch.float, copy=False) / 255.  out = F.**normalize**(out, mean, std)  out = np.**transpose**(out.**numpy**(), axes=[1, 2, 0])  **return** out  **make\_calib\_man**(  pre\_ftn=preprocess\_resnet50, *# callable function to pre-process the calibration data*  data\_dir="/datasets/imagenet/cali\_1000", *# path to folder of original calibration data files such as images*  save\_dir="/workspace/calibration/", *# path to folder to save pre-proceessed calibration data files*  save\_name="resnet50", *# tag for the generated calibration dataset*  max\_size=50 *# Maximum number of data to use for calibration*  ) |

The above results are in a directory containing the pre-processed calibration dataset (numpy tensors), located at `/workspace/sample/calibration/resnet50`. In addition, a calibration meta txt file containing the paths to the pre-processed numpy files is created, named `/workspace/sample/calibration/resnet50.txt`.

**Remark** Unless the custom pre-processing function contains proper exception handling, the sample dataset for calibration should be composed of images with the same format. Like the previous method, the calibration dataset will not be created properly if some are in color images and others are in grayscale images.

### Use Mobilint® Calibration GUI Tool

[Mobilint® Calibration GUI](http://git.mobilint/AlgorithmGroup/Calibration_GUI) is a tool that helps users to make the calibration dataset. With a prepared dataset of image files, users can easily generate a pre-processed calibration dataset of npy files. The tool provides pre-defined pre-processing functions for various deep learning models.



1. Figure 4‑1. Mobilint® Calibration GUI

## Compiling ONNX Models

ONNX is a recommended framework to be used for compiling the trained model. With simple code, the ONNX model can be directly parsed to obtain Mobilint IR.  example code is shown below. The following code assumes that the calibration dataset and the model are prepared in the directory `/workspace/calibration/resnet50` and `/workspace/resnet50.onnx`, respectively.

|  |
| --- |
| *""" Compile ONNX model"""*  **from** qubee **import** mxq\_compile  onnx\_model\_path = "/workspace/resnet50.onnx"  calib\_data\_path = "/workspace/calibration/resnet50"  *# calib\_data\_path can be replaced with the path to the calibration meta file such as "/workspace/calibration/resnet50.txt"*  **mxq\_compile**(  model=onnx\_model\_path,  calib\_data\_path=calib\_data\_path,  save\_path="resnet50.mxq",  backend="onnx"  ) |

## Compiling PyTorch Models

PyTorch models can be compiled in two different ways. The first approach is converting the PyTorch model into the ONNX model with [`torch.onnx`](https://pytorch.org/docs/stable/onnx.html) namespace, and compiling the converted model with the ONNX backend. The second approach is directly plugging the model into the Mobilint IR. Once the model is converted to Mobilint IR, then it is compiled into MXQ. The example code is shown below. The following codes assume that the calibration dataset is prepared in the directory `/workspace/calibration/resnet50`.

|  |
| --- |
| *""" Compile PyTorch model"""*  **from** qubee **import** mxq\_compile  *### get resnet50 from torchvision*  **import** torchvision  **import** torch  calib\_data\_path = "/workspace/calibration/resnet50"  *# A calibration meta file such as "/workspace/calibration/resnet50.txt" can be used instead.*  *### get resnet50 from torchvision and convert it to torchscript*  torch\_model = torchvision.models.**resnet50**(pretrained=True)  torchscript\_model\_path = "/workspace/resnet50.pt"  example\_input = torch.**rand**(1, 3, 224, 224)  scripted\_model = torch.jit.**script**(torch\_model, example\_input)  torch.jit.**save**(scripted\_model, torchscript\_model\_path)  **mxq\_compile**(  model=torchscript\_model\_path,  calib\_data\_path=calib\_data\_path,  backend="torchscript",  save\_path="resnet50.mxq",  example\_input=example\_input  ) |

## Compiling TensorFlow/Keras Models

Since Keras works as an interface for TensorFlow, models on the Keras framework can be converted to Mobilint IR via TensorFlow. Currently, the qubee compiler supports TensorFlow models saved in the format of the SavedModel or frozen graph. For the SavedModel format, which includes the serialized model ending with `.pb`, the model can be directly compiled. For the frozen graph, the compiler requires the input node name and the output node name, which can be viewed by [Netron](https://netron.app/). The following codes assume the calibration dataset is prepared in the directory `/workspace/calibration/resnet50`.

|  |
| --- |
| *""" Compile Keras/TensorFlow model in SavedModel format """*  **from** qubee **import** mxq\_compile  **import** tensorflow as tf  keras\_model = tf.keras.applications.resnet50.**ResNet50**() *# Load a Keras model*  calib\_data\_path = "/workspace/calibration/resnet50"  *# A calibration metadata file such as "/workspace/calibration/resnet50.txt" can be used instead.*  keras\_model\_save\_path = "/workspace/tf\_models/resnet50" *# directory to save the model*  keras\_model.**save**(keras\_model\_save\_path) *# Save the model in the format of the frozen graph. saved\_model.pb file will be created in the directory.*  **mxq\_compile**(      model=keras\_model\_save\_path,      calib\_data\_path=calib\_data\_path,      backend="tf",      save\_path="resnet50.mxq",  ) |

To test the model in the format of the frozen graph, download `MobileNet\_v2\_1.0\_224` from the [TensorFlow Model Garden](https://github.com/tensorflow/models/tree/master/research/slim) and save it in the directory `/workspace/tf\_models/mobilenet\_v2`. The following code assumes the calibration dataset is prepared in the directory `/workspace/calibration/resnet50`, which contains the calibration data that is compatible with the model `MobileNet`.

**Remark** Compiling the model in the frozen graph format requires the input and output node name. When the input tensor name is `input`, it is recommended to set the input node name as `input:0`. In the case that the input tensor shape is unknown, the input shape should be set as `input:0[-1,224,224,3]`, where -1 indicates the batch dimension, and `[224,224,3]` is the input shape. The output argument should be set in the same way.

|  |
| --- |
| *""" Compile TensorFlow model in frozen graph format """*  **from** qubee **import** mxq\_compile  **import** tensorflow as tf  calib\_data\_path = "/workspace/calibration/resnet50"  *# A calibration metadata file such as "/workspace/calibration/resnet50.txt" can be used instead.*  tf\_model\_save\_path = "/workspace/tf\_models/mobilenet\_v2/mobilenet\_v2\_1.0\_224\_frozen.pb" *# directory to save the model*  feed\_dict = {"input": ["input:0[-1,224,224,3]"], "output": ["MobilenetV2/Predictions/Softmax:0"]}  **mxq\_compile**(      model=tf\_model\_save\_path,      calib\_data\_path=calib\_data\_path,      backend="tf",      save\_path="mobilenet\_v2.mxq",      feed\_dict=feed\_dict  ) |

## Compiling TensorFlow Lite Models

The qubee compiler supports TensorFlow Lite models. With the given TensorFlow Lite model, the calibration dataset, and the backend, the model can be compiled into Mobilint IR. The following code assumes the calibration dataset is prepared in the directory `/workspace/calibration/resnet50`.

|  |
| --- |
| *""" Compile Tensorflow Lite model """*  **from** qubee **import** mxq\_compile  **import** tensorflow as tf  keras\_model = tf.keras.applications.resnet50.**ResNet50**() *# Load a pre-trained Keras model*  input\_shape = (224, 224, 3)  calib\_data\_path = "/workspace/calibration/resnet50"  *# A calibration metadata file such as "/workspace/calibration/resnet50.txt" can be used instead.*  keras\_model\_save\_path = "/workspace/tf\_models/resnet50" *# directory to save the Tensorflow model*  keras\_model.**save**(keras\_model\_save\_path) *# Save the model in the format of the frozen graph. saved\_model.pb file will be created in the directory.*  tflite\_model = tf.lite.TFLiteConverter.**from\_saved\_model**(keras\_model\_save\_path).**convert**() *# Convert the model to TFLite format*  with **open**('/workspace/tf\_models/resnet50.tflite', 'wb') as f:      f.**write**(tflite\_model)  **mxq\_compile**(      model=keras\_model\_save\_path+".tflite",      calib\_data\_path=calib\_data\_path,      backend="tflite",      save\_path="resnet50.mxq",  ) |

## Compling Models with Custom Input(ONNX/Torch/TensorFlow Frameworks)

When the model lacks input shape information, qubee may generate the following error:

|  |
| --- |
| ValueError: Input node <node name> has more than one unknown shape. Please enter the numpy input array to infer the input shape. |

If you encounter this error, you should provide numpy input arrays along with the model during compilation. Ensure the folder structure is as follows:

|  |
| --- |
| - <folder name>  |- <input node name 1>.npy // only **for** input node whose shape is unknown.  |- ...  |- <input node name n>.npy |

For example, if your model has three inputs named `<input1, input2, input3>` and the shape of `input2` and `input3` are unknown, then you should prepare the numpy array for `input2` and `input3` with the following folder structure.

|  |
| --- |
| - custom\_input\_array  |- input2.npy  |- input3.npy |

With the above array, you can compile the model as follows:

|  |
| --- |
| *# compile\_test.py*  **import** argparse    **from** qubee **import** mxq\_compile  **from** qubee.utils.utils\_model\_dict **import** parse\_custom\_input\_info    onnx\_model\_path = "/workspace/deeplabv3\_mobilenet\_v3\_large\_torchvision.onnx"  calib\_data\_path = "/workspace/calibration/deeplabv3"    parser = argparse.**ArgumentParser**(description="Compile arguments")  parser.**add\_argument**("--input\_shape\_path", dest="custom\_input\_shape\_dict", action=parse\_custom\_input\_info)      **mxq\_compile**(      model=onnx\_model\_path,      calib\_data\_path=calib\_data\_path,      save\_path="deeplabv3.mxq",      input\_shape\_dict=args.custom\_input\_shape\_dict      backend="onnx"  ) |

Then, you can compile the model with the following command:

|  |
| --- |
| python compile\_test.py --input\_shape\_path /workspace/custom\_input\_array |

# CPU Offloading (Beta Version)

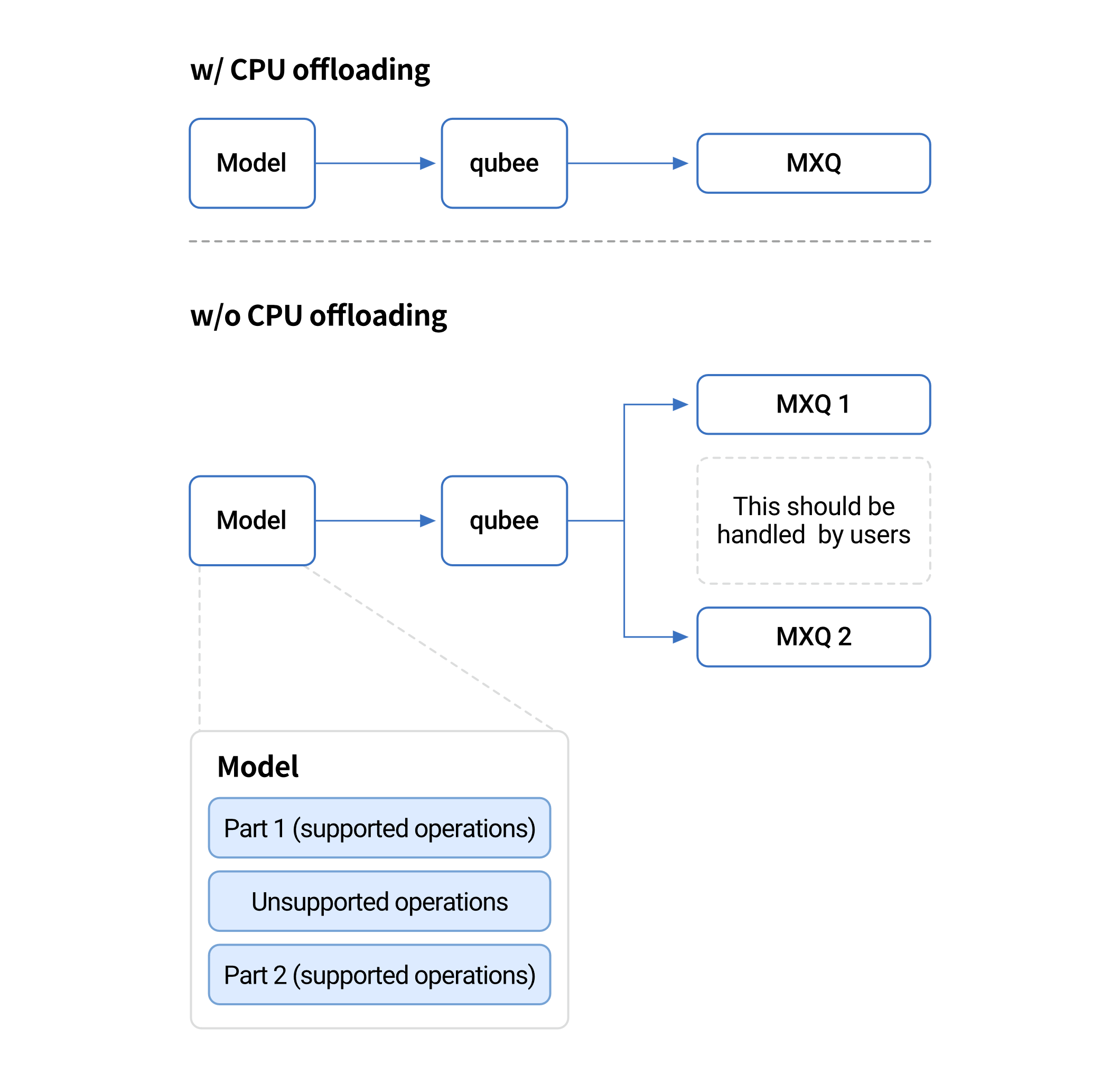
**Remark** To proceed inference with CPU offloading, it requires a runtime library that supports the MXQ file that is compiled for CPU offloading.

From qubee v0.7, we provide a Beta version of CPU offloading for mxq compile. CPU offloading makes it easier for users to compile their models by automatically offloading the computation that Mobilint NPU does not support to the CPU. For example, if a pre-processing or post-processing included in the model involves operations that the NPU does not support, the user would have to implement them manually after compiling, but CPU offloading covers most of these operations and eliminates the need for additional work.

When CPU offloading is employed, the procedures for preparing the calibration dataset and compiling the model vary slightly as follows:

(i) The pre-processed input shape should match the original model's input shape, whereas the pre-processed input shape should be in the format (H, W, C) to compile the model without CPU offloading.

(ii) Set the argument *cpu\_offload* of function *mxq\_compile* True to enable CPU offloading.



1. Figure 5‑1. SDK CPU Offloading

# Supported Frameworks

We support almost all the commonly used Machine Learning frameworks & libraries such as ONNX, PyTorch, Keras, TensorFlow, and TensorFlow Lite.



1. Figure 6‑1. Supported deep-learning frameworks

## Supported Operations (ONNX)

Table 6‑1. ONNX Supported Operations

| API Name | Comments |
| --- | --- |
| [Add](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Add) | Broadcast only for specific cases of constant addition:  Adding scalar,  Adding channel-size vector. |
| [And](https://github.com/onnx/onnx/blob/main/docs/Operators.md#And) |  |
| [ArgMax](https://github.com/onnx/onnx/blob/main/docs/Operators.md#ArgMax) |  |
| [AveragePool](https://github.com/onnx/onnx/blob/main/docs/Operators.md#AveragePool) | Only dilation=1, count\_include\_pad=1. |
| [BatchNormalization](https://github.com/onnx/onnx/blob/main/docs/Operators.md#BatchNormalization) | Only training\_mode=0. |
| [Cast](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Cast) |  |
| [Ceil](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Ceil) |  |
| [Clip](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Clip) |  |
| [Concat](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Concat) | Only along channel axis. |
| [Constant](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Constant) |  |
| [ConstantOfShape](https://github.com/onnx/onnx/blob/main/docs/Operators.md#ConstantOfShape) |  |
| [Conv](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Conv) |  |
| [ConvTranspose](https://github.com/onnx/onnx/blob/main/docs/Operators.md#ConvTranspose) |  |
| [DepthToSpace](https://github.com/onnx/onnx/blob/main/docs/Operators.md#DepthToSpace) |  |
| [Div](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Div) | Only constant division.  Support broadcast same as Add. |
| [Elu](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Elu) |  |
| [Equal](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Equal) |  |
| [Erf](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Erf) |  |
| [Exp](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Exp) |  |
| [Expand](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Expand) |  |
| [Flatten](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Flatten) | Only axis=1 and before fully connected layer or Conv w/ 1x1 kernel. |
| [Floor](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Floor) |  |
| [Gather](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Gather) |  |
| [GatherND](https://github.com/onnx/onnx/blob/main/docs/Operators.md#GatherND) |  |
| [Gemm](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Gemm) | Only transA=0.  Only for the following specific case:  Input A is a flatten activation and input B is 2D tensor. |
| [GlobalAveragePool](https://github.com/onnx/onnx/blob/main/docs/Operators.md#GlobalAveragePool) |  |
| [Greater](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Greater) |  |
| [HardSigmoid](https://github.com/onnx/onnx/blob/main/docs/Operators.md#HardSigmoid) |  |
| [HardSwish](https://github.com/onnx/onnx/blob/main/docs/Operators.md#HardSwish) |  |
| [Identity](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Identity) |  |
| [InstanceNormalization](https://github.com/onnx/onnx/blob/main/docs/Operators.md#InstanceNormalization) |  |
| [LayerNormalization](https://github.com/onnx/onnx/blob/main/docs/Operators.md#LayerNormalization) |  |
| [LeakyRelu](https://github.com/onnx/onnx/blob/main/docs/Operators.md#LeakyRelu) |  |
| [Less](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Less) |  |
| [Loop](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Loop) |  |
| [MatMul](https://github.com/onnx/onnx/blob/main/docs/Operators.md#MatMul) | Only for the following specific case:  Input A is a flatten activation and input B is 2D tensor or vice-versa. |
| [Max](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Max) |  |
| [MaxPool](https://github.com/onnx/onnx/blob/main/docs/Operators.md#MaxPool) | Only dilation=1. |
| [Min](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Min) |  |
| [Mod](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Mod) |  |
| [Mul](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Mul) | Only constant multiplication.  Support broadcast same as Add. |
| [NonMaxSuppression](https://github.com/onnx/onnx/blob/main/docs/Operators.md#NonMaxSuppression) |  |
| [NonZero](https://github.com/onnx/onnx/blob/main/docs/Operators.md#NonZero) |  |
| [Not](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Not) |  |
| [Or](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Or) |  |
| [Pad](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Pad) | Constant, reflect, edge modes are supported |
| [Pow](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Pow) |  |
| [PRelu](https://github.com/onnx/onnx/blob/main/docs/Operators.md#PRelu) |  |
| [Range](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Range) |  |
| [Reciprocal](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Reciprocal) |  |
| [ReduceMax](https://github.com/onnx/onnx/blob/main/docs/Operators.md#ReduceMax) | Only along height and width. |
| [ReduceMean](https://github.com/onnx/onnx/blob/main/docs/Operators.md#ReduceMean) | Only along height and width. |
| [ReduceMin](https://github.com/onnx/onnx/blob/main/docs/Operators.md#ReduceMin) | Only along height and width. |
| [ReduceProd](https://github.com/onnx/onnx/blob/main/docs/Operators.md#ReduceProd) | Only along height and width. |
| [ReduceSum](https://github.com/onnx/onnx/blob/main/docs/Operators.md#ReduceSum) | Only along height and width. |
| [Relu](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Relu) | Only scalar slope. |
| [Reshape](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Reshape) | Only channel-wise flatten and before fully connected layer or Conv w/ 1x1 kernel.  Only allowzero=0. |
| [Resize](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Resize) | Only for the following specific case:  Only mode = "nearest" and coordinate\_transformation\_mode = "half\_pixel" or "pytorch\_half\_pixel",  Only mode = "linear" and coordinate\_transformation\_mode = "half\_pixel" or "pytorch\_half\_pixel",  Attributes axes, antialias, keep\_aspect\_ratio\_policy nearest\_mode are not supported. |
| [ScatterND](https://github.com/onnx/onnx/blob/main/docs/Operators.md#ScatterND) |  |
| [Shape](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Shape) |  |
| [Sigmoid](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Sigmoid) |  |
| [Slice](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Slice) | Only channel-wise slice. |
| [Softmax](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Softmax) |  |
| [Softplus](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Softplus) |  |
| [Split](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Split) |  |
| [Sqrt](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Sqrt) |  |
| [Squeeze](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Squeeze) | Only when resulting tensor has 2D shape.  Squeeze along batch axis is unsupported. |
| [Sub](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Sub) | Support broadcast same as Add. |
| [Tanh](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Tanh) |  |
| [Tile](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Tile) | Batch-wise tile is unsupported. |
| [TopK](https://github.com/onnx/onnx/blob/main/docs/Operators.md#TopK) |  |
| [Transpose](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Transpose) | Only for the following specific case:  Transpose-Flatten-Linear. |
| [Unsqueeze](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Unsqueeze) |  |
| [Upsample](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Upsample) | Only mode "nearest" and "linear". |
| [Xor](https://github.com/onnx/onnx/blob/main/docs/Operators.md#Xor) |  |

## Supported operations (PyTorch)

**Remark** Since the Torchscript backend framework is based on [Torchscript-Based-ONNX-Exporter](https://pytorch.org/docs/stable/onnx.html#Torchscript-Based-ONNX-Exporter), even if the operation is not listed below, it may be supported if it has corresponding ONNX operation, which is supported by qubee.

Table 6‑2. PyTorch Supported Operations

| API Name | Comments |
| --- | --- |
| [ADD](https://pytorch.org/docs/1.13/generated/torch.add.html#torch.add) | Only alpha=1.  See supported operations (ONNX): Add |
| [AMAX](https://pytorch.org/docs/1.13/generated/torch.amax.html#torch.amax) | See supported operations (ONNX): ReduceMax |
| [AMIN](https://pytorch.org/docs/1.13/generated/torch.amin.html#torch.amin) | See supported operations (ONNX): ReduceMin |
| [ARGMAX](https://pytorch.org/docs/1.13/generated/torch.argmax.html#torch.argmax) | See supported operations (ONNX): ArgMax |
| [CAT](https://pytorch.org/docs/1.13/generated/torch.cat.html#torch.cat) | Only along channel axis.  See supported operations (ONNX): Concat |
| [CEIL](https://pytorch.org/docs/1.13/generated/torch.ceil.html#torch.ceil) | See supported operaitons (ONNX): Ceil |
| [CLAMP](https://pytorch.org/docs/1.13/generated/torch.clamp.html#torch.clamp) | See supported operations (ONNX): Clip |
| [DIV](https://pytorch.org/docs/1.13/generated/torch.div.html#torch.div) | Only constant division.  See supported operations (ONNX): Div |
| [EQ](https://pytorch.org/docs/1.13/generated/torch.eq.html#torch.eq) | See supported operations (ONNX): Equal |
| [ERF](https://pytorch.org/docs/1.13/generated/torch.erf.html#torch.erf) | See supported operations (ONNX): Erf |
| [EXP](https://pytorch.org/docs/1.13/generated/torch.exp.html#torch.exp) | See supported operations (ONNX): Exp |
| [FLOOR](https://pytorch.org/docs/1.13/generated/torch.floor.html#torch.floor) | See supported operations (ONNX): Floor |
| [FMOD](https://pytorch.org/docs/1.13/generated/torch.fmod.html#torch.fmod) | See supported operations (ONNX): Mod |
| [GATHER](https://pytorch.org/docs/1.13/generated/torch.gather.html#torch.gather) | See supported operations (ONNX): GatherND |
| [GT](https://pytorch.org/docs/1.13/generated/torch.gt.html#torch.gt) | See supported operations (ONNX): Greater |
| [LOGICAL\_AND](https://pytorch.org/docs/1.13/generated/torch.logical_and.html#torch.logical_and) | See supported operations (ONNX): And |
| [LOGICAL\_NOT](https://pytorch.org/docs/1.13/generated/torch.logical_not.html#torch.logical_not) | See supported operations (ONNX): Not |
| [LOGICAL\_OR](https://pytorch.org/docs/1.13/generated/torch.logical_or.html#torch.logical_or) | See supported operations (ONNX): Or |
| [LOGICAL\_XOR](https://pytorch.org/docs/1.13/generated/torch.logical_xor.html#torch.logical_xor) | See supported operations (ONNX): Xor |
| [LT](https://pytorch.org/docs/1.13/generated/torch.lt.html#torch.lt) | See supported operations (ONNX): Less |
| [MATMUL](https://pytorch.org/docs/1.13/generated/torch.matmul.html#torch.matmul) | See supported operations (ONNX): MatMul |
| [MAX](https://pytorch.org/docs/1.13/generated/torch.max.html#torch.max) | See supported operations (ONNX): Max |
| [MEAN](https://pytorch.org/docs/1.13/generated/torch.mean.html#torch.mean) | See supported operations (ONNX): ReduceMean |
| [MIN](https://pytorch.org/docs/1.13/generated/torch.min.html#torch.min) | See supported operations (ONNX): Min |
| [MUL](https://pytorch.org/docs/1.13/generated/torch.mul.html#torch.mul) | Only constant multiplication.  See supported operations (ONNX): Mul |
| [ADAPTIVEAVGPOOL2D](https://pytorch.org/docs/1.13/generated/torch.nn.AdaptiveAvgPool2d.html#torch.nn.AdaptiveAvgPool2d) | See supported operations (ONNX): AveragePool |
| [ADAPTIVEMAXPOOL2D](https://pytorch.org/docs/1.13/generated/torch.nn.AdaptiveMaxPool2d.html#torch.nn.AdaptiveMaxPool2d) | See supported operations (ONNX): MaxPool |
| [AVGPOOL2D](https://pytorch.org/docs/1.13/generated/torch.nn.AvgPool2d.html#torch.nn.AvgPool2d) | Only dilation=1, count\_include\_pad=1.  See supported operations (ONNX): AveragePool |
| [BATCHNORM2D](https://pytorch.org/docs/1.13/generated/torch.nn.BatchNorm2d.html#torch.nn.BatchNorm2d) | See supported operations (ONNX): BatchNormalization |
| [CONV2D](https://pytorch.org/docs/1.13/generated/torch.nn.Conv2d.html#torch.nn.Conv2d) | See supported operations (ONNX): Conv |
| [CONVTRANSPOSE2D](https://pytorch.org/docs/1.13/generated/torch.nn.ConvTranspose2d.html#torch.nn.ConvTranspose2d) | See supported operations (ONNX): ConvTranspose |
| [ELU](https://pytorch.org/docs/1.13/generated/torch.nn.ELU.html#torch.nn.ELU) | See supported operations (ONNX): Elu |
| [FLATTEN](https://pytorch.org/docs/1.13/generated/torch.nn.Flatten.html#torch.nn.Flatten) | Only channel-wise flatten and before fully connected layer or Conv w/ 1x1 kernel.  See supported operations (ONNX): Flatten |
| [INTERPOLATE](https://pytorch.org/docs/1.13/generated/torch.nn.functional.interpolate.html#torch.nn.functional.interpolate) | See supported operations (ONNX): Resize. |
| [PAD](https://pytorch.org/docs/1.13/generated/torch.nn.functional.pad.html#torch.nn.functional.pad) | See supported operations (ONNX): Pad |
| [HARDSIGMOID](https://pytorch.org/docs/1.13/generated/torch.nn.Hardsigmoid.html#torch.nn.Hardsigmoid) | See supported operations (ONNX): HardSigmoid |
| [HARDSWISH](https://pytorch.org/docs/1.13/generated/torch.nn.Hardswish.html#torch.nn.Hardswish) | See supported operations (ONNX): HardSwish |
| [IDENTITY](https://pytorch.org/docs/1.13/generated/torch.nn.Identity.html#torch.nn.Identity) | See supported operations (ONNX): Identity |
| [INSTANCENORM2D](https://pytorch.org/docs/1.13/generated/torch.nn.InstanceNorm2d.html#torch.nn.InstanceNorm2d) | See supported operations (ONNX): InstanceNormalization |
| [LEAKYRELU](https://pytorch.org/docs/1.13/generated/torch.nn.LeakyReLU.html#torch.nn.LeakyReLU) | See supported operations (ONNX): LeakyRelu |
| [LINEAR](https://pytorch.org/docs/1.13/generated/torch.nn.Linear.html#torch.nn.Linear) | See supported operations (ONNX): Gemm |
| [MAXPOOL2D](https://pytorch.org/docs/1.13/generated/torch.nn.MaxPool2d.html#torch.nn.MaxPool2d) | Only dilation=1.  See supported operations (ONNX): MaxPool |
| [PRELU](https://pytorch.org/docs/1.13/generated/torch.nn.PReLU.html#torch.nn.PReLU) | See supported operations (ONNX): PRelu |
| [RELU](https://pytorch.org/docs/1.13/generated/torch.nn.ReLU.html#torch.nn.ReLU) | See supported operations (ONNX): Relu |
| [SIGMOID](https://pytorch.org/docs/1.13/generated/torch.nn.Sigmoid.html#torch.nn.Sigmoid) | See supported operations (ONNX): Sigmoid |
| [SOFTMAX](https://pytorch.org/docs/1.13/generated/torch.nn.Softmax.html#torch.nn.Softmax) | See supported operations (ONNX): Softmax |
| [SOFTPLUS](https://pytorch.org/docs/1.13/generated/torch.nn.Softplus.html#torch.nn.Softplus) | Only beta=1.  See supported operations (ONNX): Softplus |
| [TANH](https://pytorch.org/docs/1.13/generated/torch.nn.Tanh.html#torch.nn.Tanh) | See supported operations (ONNX): Tanh |
| [UPSAMPLE](https://pytorch.org/docs/1.13/generated/torch.nn.Upsample.html#torch.nn.Upsample) | Only mode "nearest" and "linear".  Only scales=[2,2].  See supported operations (ONNX): Upsample |
| [PERMUTE](https://pytorch.org/docs/1.13/generated/torch.permute.html#torch.permute) | See supported operations (ONNX): Transpose |
| [POW](https://pytorch.org/docs/1.13/generated/torch.pow.html#torch.pow) | See supported operations (ONNX): Pow |
| [PROD](https://pytorch.org/docs/1.13/generated/torch.prod.html#torch.prod) | See supported operations (ONNX): ReduceProd |
| [RECIPROCAL](https://pytorch.org/docs/1.13/generated/torch.reciprocal.html#torch.reciprocal) | See supported operations (ONNX): Reciprocal |
| [RESHAPE](https://pytorch.org/docs/1.13/generated/torch.reshape.html#torch.reshape) | Only channel-wise flatten and before fully connected layer or Conv w/ 1x1 kernel.  See supported operations (ONNX): Reshape |
| [SCATTER](https://pytorch.org/docs/1.13/generated/torch.scatter.html#torch.scatter) | See supported operaitons (ONNX): ScatterND |
| [SPLIT](https://pytorch.org/docs/1.13/generated/torch.split.html#torch.split) | See supported operations (ONNX): Split |
| [SQRT](https://pytorch.org/docs/1.13/generated/torch.sqrt.html#torch.sqrt) | See supported operations (ONNX): Sqrt |
| [SQUEEZE](https://pytorch.org/docs/1.13/generated/torch.squeeze.html#torch.squeeze) | Only when resulting tensor has 2D shape.  Squeeze along batch axis is unsupported.  See supported operations (ONNX): Squeeze |
| [SUB](https://pytorch.org/docs/1.13/generated/torch.sub.html#torch.sub) | Only alpha=1.  See supported operations (ONNX): Sub |
| [TENSOR](https://pytorch.org/docs/1.13/generated/torch.tensor.html#torch.tensor) | See supported operations (ONNX): Constant |
| [TILE](https://pytorch.org/docs/1.13/generated/torch.tile.html#torch.tile) | Batch-wise tile is unsupported.  See supported operations (ONNX): Tile |
| [TOPK](https://pytorch.org/docs/1.13/generated/torch.topk.html#torch.topk) | See supported operations (ONNX): TopK |
| [TRANSPOSE](https://pytorch.org/docs/1.13/generated/torch.transpose.html#torch.transpose) | Only before fully connected layer.  See supported operations (ONNX): Transpose |
| [UNSQUEEZE](https://pytorch.org/docs/1.13/generated/torch.unsqueeze.html#torch.unsqueeze) | See supported operations (ONNX): Unsqueeze |

## Supported operations (TensorFlow/Keras/TensorFlow Lite)

As mentioned in the previous section, Keras works as an interface for TensorFlow 2, and they save the model in the same format as the frozen graph, which ends with `.pb`. Therefore, the TensorFlow/Keras/TensorflowLite operation is supported if it can be described by the TensorFlow raw operations listed below when the model is saved in the format of the frozen graph.

Table 6‑3. TensorFlow Supported Operations

| API Name | Comments |
| --- | --- |
| [Placeholder](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Placeholder) | According to the official document, this operation will fail with an error if it is executed. |
| [PlaceholderWithDefault](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/PlaceholderWithDefault) |  |
| [Floor](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Floor) |  |
| [Identity](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Identity) |  |
| [Const](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Const) |  |
| [IdentityN](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/IdentityN) |  |
| [Pad](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Pad) |  |
| [PadV2](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/PadV2) |  |
| [Conv2D](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Conv2D) |  |
| [DepthwiseConv2dNative](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/DepthwiseConv2dNative) |  |
| [Conv2DBackpropInput](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Conv2DBackpropInput) |  |
| [MatMul](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/MatMul) |  |
| [FusedBatchNormV3](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/FusedBatchNormV3) |  |
| [FusedBatchNorm](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/FusedBatchNorm) |  |
| [MaxPool](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/MaxPool) |  |
| [AvgPool](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/AvgPool) |  |
| [Mean](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Mean) | Only along height, width, and channel. |
| [ResizeNearestNeighbor](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/ResizeNearestNeighbor) |  |
| [ResizeBilinear](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/ResizeBilinear) |  |
| [ConcatV2](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/ConcatV2) | Only along channel axis. |
| [Add](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Add) |  |
| [AddV2](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/AddV2) |  |
| [Mul](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Mul) | Only constant multiplication. |
| [Sub](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Sub) |  |
| [RealDiv](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/RealDiv) | Only constant division. |
| [AddN](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/AddN) |  |
| [BiasAdd](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/BiasAdd) |  |
| [Relu](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Relu) |  |
| [Sigmoid](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Sigmoid) |  |
| [Softplus](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Softplus) |  |
| [Exp](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Exp) |  |
| [Tanh](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Tanh) |  |
| [Neg](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Neg) |  |
| [LeakyRelu](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/LeakyRelu) |  |
| [Relu6](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Relu6) |  |
| [Softmax](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Softmax) |  |
| [ArgMax](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/ArgMax) |  |
| [Switch](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Switch) |  |
| [Merge](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Merge) |  |
| [Shape](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Shape) |  |
| [Reshape](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Reshape) | Only channel-wise flatten and before fully connected layer or Conv w/ 1x1 kernel. |
| [Transpose](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Transpose) | Only before fully connected layer. |
| [ExpandDims](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/ExpandDims) |  |
| [Squeeze](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Squeeze) | Only when resulting tensor has 2D shape.  Squeeze along batch axis is unsupported. |
| [StridedSlice](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/StridedSlice) | ellipsis\_mask, new\_axis\_mask, shrink\_axis\_mask are unsupported. |
| [Slice](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Slice) |  |
| [Pack](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Pack) |  |
| [Split](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Split) |  |
| [SplitV](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/SplitV) |  |
| [Range](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Range) |  |
| [Fill](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Fill) |  |
| [Tile](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Tile) | Batch-wise tile is unsupported. |
| [Cast](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Cast) |  |
| [TensorArrayV3](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/TensorArrayV3) |  |
| [Maximum](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Maximum) |  |
| [Sqrt](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Sqrt) |  |
| [Rsqrt](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Rsqrt) |  |
| [Rint](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Rint) |  |
| [Greater](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Greater) |  |
| [LogicalAnd](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/LogicalAnd) |  |
| [Equal](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Equal) |  |
| [GreaterEqual](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/GreaterEqual) |  |
| [RandomUniform](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/RandomUniform) |  |
| [NoOp](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/NoOp) |  |
| [Assert](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/Assert) |  |
| [ReadFile](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/ReadFile) |  |
| [DecodeJpeg](https://www.tensorflow.org/versions/r2.9/api_docs/python/tf/raw_ops/DecodeJpeg) |  |

# API Reference

## Function: mxq\_compile

Compile a given model directly without creating an instance of "Model\_Dict".

Table 7‑1. mxq\_compile

| Parameter | Type | Description |
| --- | --- | --- |
| model | string or model instance | Model path or model instance. Model should be instance for the following cases:  When using backend="onnx", it should be the path to ONNX model file  When using backend="torchscript", it should be the path to PyTorch model file  When using backend="tf", it should be the path to the folder saving TensorFlow PB graph and assets.  When using backend="tflite", it should be the path to TF Lite model file. |
| calib\_data\_path | string | A path to the calibration dataset. It can be either of a path to the text (or json) file containing the paths to the pre-processed numpy files or a directory containing the pre-processed numpy files. |
| model\_nickname | string (optional) | Model nickname used in qubee. It is used in qubee to facilitate quicker recompilation of the same models.  Qubee stores prior optimization information under this nickname, enabling it to locate and utilize the previously compiled results for faster processing.  It is auto-generated from the model's base name, if not provided.  For instance, a model "/workspace/onnx/resnet50.onnx" results in "resnet50".  If not derivable, "temporary" is the default nickname. |
| save\_path | string (optional) | Filename of the resulting .mxq.  If it is None, then it is set to "model\_nickname".mxq  Defaults to None. |
| input\_shape | tuple or list (optional) | Input shape in HWC. Required only for using PyTorch model and backend="torchscript". |
| backend | string (optional) | Which framework to use to get the Mobilint IR.  It must be one of "onnx", "tf1", "tf2", and "torchscript".  They correspond to deep learning frameworks as follows:  "onnx": ONNX,  "tf": TensorFlow,  "tflite": TensorFlow Lite,  "torchscript": PyTorch  Defaults to "onnx". |
| device | string (optional) | Device to be used for compile and inerence. Either "cpu" or "gpu".  Defaults to "cpu". |
| quantize\_method | string (optional) | Quantization method to determine the scale parameter in the quantization.  Currently, "Max", "Percentile", "MaxPercentile" and "KL" are supported.  Defaults to "Percentile". |
| quantize\_percentile | float (optional) | Percentile used for the quantization method "Percentile" and "MaxPercentile".  This should be between 0 and 1. (Ex. 0.999, 0.9999)  Defaults to 0.99995. |
| topk\_ratio | float (optional) | It is used for quantization method "maxpercentile". Defaults to 0.  The larger this value is, the more data is used for calibration.  This should be between 0 and 1, but using a value of 0.01 or less is recommended. |
| smooth\_factor | float (optional) | Smooth factor for Gaussian kernel construction, which is required on KL divergence estimation.  Defaults to 1.6. |
| is\_quant\_ch | bool (optional) | Use multi-channel quantization if True. Defaults to False. |
| optimization | bool (optional) | If True, it compiles the model with optimization process. If false, qubee uses  previous optimization information when stored in previous compiling.  (Nickname should be the same.) It must be set to True on the first compile.  Defaults to True. |
| optimization\_level | int (optional) | Optimization level in the compiler. If optimization level is high, NPU inference  could be faster, but it takes more time for compiling. (Recommend: 3~6)  Defaults to 5. |
| save\_sample | bool (optional) | If True, create the "sampleInOut" folder in the current directory and store the input and output binary files in it.  Defaults to False. |
| use\_random\_calib | bool (optional) | If True, it compiles the given model with random calibration data.  This is just used to check if the model is compilable without making a calibration data.  Defaults to False. |
| cpu\_offload | bool (optional) | Use CPU offloading for NPU inference if True. Defaults to False. |
| quant\_output | string (optional) | Quantization method that applied to the output layer. "layer", "ch" and "sigmoid" options are available.  If "layer", per-layer quantization is applied to the output layer. If is\_quant\_ch is true, then the computed quantization scale for each channel of the output layer will be merged into single value.  If "ch", per-channel quantization is applied to the output layer. This option is valid only when is\_quant\_ch is true.  If "sigmoid", assign quantization scale that computed with sigmoid function.  Defaults to "layer". |
| adaq\_useadaquant | bool (optional) | If True, enable the finetuning with AdaQuant after quantization.  Defaults to False. |
| adaq\_weightDeltaLR | float (optional) | Learning rate for finetuning weight delta(weight update) of AdaQuant. (Recommend: 1e-6 ~ 5e-5)  Defaults to 0. |
| adaq\_biasDeltaLR | float (optional) | Learning rate for finetuning bias delta(bias update) of AdaQuant. (Recommend: weightDeltaLR/10 ~ weightDeltaLR/2)  Defaults to 0. |
| adaq\_weightScaleLR | float (optional) | Learning rate for finetuning weight quantization scale of AdaQuant.  Defaults to 0. |
| adaq\_biasScaleLR | float (optional) | Learning rate for finetuning bias quantization scale of AdaQuant.  Defaults to 0. |
| adaq\_actScaleLR | float (optional) | Learning rate for finetuning activation quantization scale of AdaQuant.  Defaults to 0. |
| adaq\_batchSize | int (optional) | Batch size for running AdaQuant.  Defaults to 16. |
| adaq\_epoch | int (optional) | Epochs for repeating AdaQuant update.  Defaults to 10. |

### Tips for choosing quantization methods

"Percentile" and "MaxPercentile" quantization methods each take a hyperparameter called *percentile*. An increase in this value corresponds to a broader quantization interval. To elaborate further, a higher *percentile* results in reduced overflow, albeit at the expense of accuracy.

The "MaxPercentile" method determines the percentile value from data that has been filtered once. As a result, a lower *percentile* is needed for "MaxPercentile" compared to the "Percentile" method. For instance, for the "Percentile" method, we suggest using a value of 0.9999 to 0.999999. For the "MaxPercentile" method, we recommend *percentile* between 0.9 and 0.9999.

The "is\_quant\_ch" argument enables channel-wise quantization. When set to True, the quantization is performed on a per-channel basis. This method is particularly useful for models, in which activations vary significantly across channels. However, it may take a longer time to compile the model.

The "quant\_output" argument is used to determine the quantization method for the output layer. When the original model's output is various across the channels, it is recommended to set "ch" to keep channel-wise quantization. Otherwise, set "layer" to quantize the output layer as a whole.

## Class: Model\_Dict

This class serves two main functions:

1. Compile

2. Inference (Note that this inference is only for testing and done by CPU or GPU.)

Table 7‑2. Model\_Dict Class

| Attributes | Type | Description |
| --- | --- | --- |
| model\_dict | ONNX\_Model\_Dict, TF\_Model\_Dict, TFLITE\_Model\_Dict | Mobilint IR, which holds information of layers in the model. |
| model\_from | string | Backend for holding information of the model. |
| output\_name\_list | List[string] | List of the keys (in model\_dict) corresponding to the output layer of the model. (It could be different from the original model, because qubee parses deep learning related operations only.) |
| model\_from | string | Deep learning framework where the input model comes from. |
| c\_model | qubee.mmc.Compiler | Low-level compiler. (defined in C++ code). It compiles Mobilint IR into MXQ format. |
| p\_model | qubee.model\_dict.Model | Model restored from Mobilint IR. This enables full-precision inference for testing. |
| is\_compiled | bool | Indicates whether the model is compiled. |
| device | string | Device to be used for compile and inerence. Either CPU or GPU. |
| has\_c\_model | bool | Indicates whether the c\_model is prepared. |
| has\_p\_model | bool | Indicates whether the p\_model is prepared. |

### Methods

Table 7‑3. Model\_Dict Methods

| Methods | Description |
| --- | --- |
| \_\_init\_\_ | Constructor of Mobilint IR model. |
| compile | Compile the given model into MXQ format. |
| inference | Floating inference with the Mobilint IR.  This can be used to check the built IR returns the same output as the model. |
| inference\_int8 | Integer inference with the compiled and quantized model.  The model must be compiled before executing this function. |
| inference\_int8\_input\_dict | Same as "inference\_int8", but get a dictionary input which has a form of {node name: node input} instead.  This can be used for models with multiple inputs. |
| cal\_ops | Return the number of add/multiplication operations in the build Mobilint IR.  This can be reduced in later optimization steps. |
| to | Set the operating device (CPU or GPU). |

### Method Details

Table 7‑4. Model\_Dict.\_\_init\_\_

| Parameter | Type | Description |
| --- | --- | --- |
| model | string or model class of the corresponding framework | Model path or model instance. The following cases are supported:  When using backend="onnx", it should be the path to ONNX model file  When using backend="torchscript", it should be the path to PyTorch model file  When using backend="tf", it should be the path to the folder saving TensorFlow PB graph and assets.  When using backend="tflite", it should be the path to TF Lite model file. |
| backend | string (optional) | Which framework to use to get the Mobilint IR.  It must be one of "onnx", "tf", "tflite", or "torchscript".  They correspond to deep learning frameworks as follows:  "onnx": ONNX,  "tf": TensorFlow,  "tflite": TensorFlow Lite,  "torchscript": PyTorch  Defaults to "onnx". |
| input\_shape | tuple or list (optional) | Input shape in HWC. Required only for using PyTorch model. |
| device | string (optional) | Device to be used for compile and inerence. Either "cpu" or "gpu".  Defaults to "cpu". |

Table 7‑5. Model\_Dict.compile

| Parameter | Type | Description |
| --- | --- | --- |
| calib\_data\_path | string | A path to the calibration dataset. It can be either of a path to the text (or json) file containing the paths to the pre-processed numpy files or a directory containing the pre-processed numpy files. |
| save\_path | string | Filename of the resulting .mxq. |
| model\_nickname | string (optional) | Model nickname used in qubee. It is used in qubee to facilitate quicker recompilation of the same models.  Qubee stores prior optimization information under this nickname, enabling it to locate and utilize the previously compiled results for faster processing.  It is auto-generated from the model's base name, if not provided.  For instance, a model "/workspace/onnx/resnet50.onnx" results in "resnet50".  If not derivable, "temporary" is the default nickname. |
| quantize\_method | string (optional) | Quantization method to determine the scale parameter in the quantization.  Currently, "Max", "Percentile", "MaxPercentile" and "KL" are supported.  Defaults to "Percentile". |
| quantize\_percentile | float (optional) | Percentile used for the quantization method "Percentile" and "MaxPercentile".  This should be between 0 and 1. (Ex. 0.999, 0.9999).  Defaults to 0.9999. |
| topk\_ratio | float (optional) | It is used for quantization method "maxpercentile". Defaults to 0.  The larger this value is, the more data is used for calibration.  This should be between 0 and 1, but using a value of 0.01 or less is recommended. |
| smooth\_factor | float (optional) | Smoothing factor that is required for Gaussian kernel construction on KL divergence estimation.  Defaults to 1.6. |
| is\_quant\_ch | bool (optional) | Use multi-channel quantization if True. Defaults to False. |
| optimization | bool (optional) | If True, it compiles the model with optimization process.  If False, qubee uses  previous optimization information when stored in previous compiling.  (Nickname should be the same.) It must be set to True on the first compile.  Defaults to True. |
| optimization\_level | int (optional) | Optimization level in the compiler.  If optimization level is high, NPU inference  could be faster, but it takes more time for compiling. (Recommend: 3~6.)  Defaults to 5. |
| save\_sample | bool (optional) | If True, create the "sampleInOut" folder in the current directory and store the input and output binary files in it.  Defaults to False. |
| use\_random\_calib | bool (optional) | If True, it compiles the given model with random calibration data.  This is just used to check if the model is compilable without making a calibration data.  Defaults to False. |
| cpu\_offload | bool (optional) | Use CPU offloading for NPU inference if True. Defaults to False. |
| quant\_output | string (optional) | Quantization method that applied to the output layer. "layer", "ch" and "sigmoid" options are available.  If "layer", per-layer quantization is applied to the output layer. If is\_quant\_ch is true, then the computed quantization scale for each channel of the output layer will be merged into single value.  If "ch", per-channel quantization is applied to the output layer. This option is valid only when is\_quant\_ch is true.  If "sigmoid", assign quantization scale that computed with sigmoid function.  Defaults to "layer". |
| adaq\_useadaquant | bool (optional) | If True, enable the finetuning with AdaQuant after quantization.  Defaults to False. |
| adaq\_weightDeltaLR | float (optional) | Learning rate for finetuning weight delta(weight update) of AdaQuant. (Recommend: 1e-6 ~ 5e-5)  Defaults to 0. |
| adaq\_biasDeltaLR | float (optional) | Learning rate for finetuning bias delta(bias update) of AdaQuant. (Recommend: weightDeltaLR/10 ~ weightDeltaLR/2)  Defaults to 0. |
| adaq\_weightScaleLR | float (optional) | Learning rate for finetuning weight quantization scale of AdaQuant.  Defaults to 0. |
| adaq\_biasScaleLR | float (optional) | Learning rate for finetuning bias quantization scale of AdaQuant.  Defaults to 0. |
| adaq\_actScaleLR | float (optional) | Learning rate for finetuning activation quantization scale of AdaQuant.  Defaults to 0. |
| adaq\_batchSize | int (optional) | Batch size for running AdaQuant.  Defaults to 16. |
| adaq\_epoch | int (optional) | Epochs for repeating AdaQuant update.  Defaults to 10. |

Table 7‑6. Model\_Dict.inference

| Parameter | Type | Description |
| --- | --- | --- |
| input\_tensor | numpy.array  torch.Tensor  Dict[string, numpy.array or torch.Tensor]  List[numpy.array or torch.Tensor] | Input tensor with layout BCHW. |
| cast\_cpu | bool (optional) | If True, enable CPU casting on full precision inference.  Defaults to False. |

Table 7‑7. Model\_Dict.inference\_int8

| Parameter | Type | Description |
| --- | --- | --- |
| input\_tensor | torch.Tensor or np.ndarray | Input tensor with layout BCHW. |

Table 7‑8. Model\_Dict.inference\_int8\_input\_dict

| Parameter | Type | Description |
| --- | --- | --- |
| input\_dict | Dict[str, torch.Tensor or np.ndarray] | Dictionary that contains input information such as {input node name: input tensor}. |

Table 7‑9. Model\_Dict.to

| Parameter | Type | Description |
| --- | --- | --- |
| device | string | Target device to use, which must be one of "cpu", "gpu", "cuda". |

## Function: make\_calib

From the given images and preprocessing configuration, create the preprocessed numpy files and a txt file containing their paths.

Table 7‑10. make\_calib

| Parameter | Type | Description |
| --- | --- | --- |
| args\_pre | string or Dict | Path to a Yaml file or dictionary containing preprocessing configuration information.  Refer to 7.4. for details. |
| data\_dir | string | Directory of data to be used for calibration. |
| save\_dir | string | Directory to save the pre-processed numpy files and txt file which contains their paths. |
| save\_name | string (optional) | Name for resulting files.  Numpy files will be saved under {save\_dir}/{save\_name}\_npy directory.  Text file will be saved in {save\_dir}/{save\_name}.txt.  If it is not provided, it is set to the basename of data\_dir. |
| anno\_json | string (optional) | Path to an annotation json file for COCO format.  When provided, make\_calib function randomly selects samples considering class balance.  Defaults to None. |
| file\_format | string (optional) | Filename format using image\_idx.  Defaults to '%012d.jpg'. |
| max\_size | int (optional) | Maximum size of the resulting calibration data.  Defaults to -1, which means no limit on the number of the calibration data. |
| remove\_npy | bool (optional) | If True, remove pre-existing numpy files.  Defaults to False. |
| seed | int (optional) | Random seed.  Defaults to 2023. |
| save\_calib\_msg | bool (optional) | If True, save calibration data dictionary as MSGpack file.  Defaults to False. |
| msg\_path | string (optional) | Path to save MSGpack file  If not provided, it automatically generate the path with dataname and number of calibration data.  Defaults to None. |

## Fuction: make\_calib\_man

From given images and manually written function that takes an image path as input, create the preprocessed numpy files and a txt file containing their paths.

Table 7‑11. make\_calib\_man

| Parameter | Type | Description |
| --- | --- | --- |
| pre\_ftn | Callable | Pre-processing function that takes an image path as input. |
| data\_dir | string | Directory of data to be used for calibration. |
| save\_dir | string | Directory to save the pre-processed numpy files and txt file which contains their paths. |
| save\_name | string (optional) | Name for resulting files.  Numpy files will be saved under {save\_dir}/{save\_name}\_npy directory.  Text file will be saved in {save\_dir}/{save\_name}.txt.  If it is not provided, it is set to the basename of data\_dir. |
| anno\_json | string (optional) | Path to an annotation json file for COCO format.  When provided, make\_calib function randomly selects samples considering class balance.  Defaults to None. |
| file\_format | string (optional) | Filename format using image\_idx.  Defaults to '%012d.jpg'. |
| max\_size | int (optional) | Maximum size of the resulting calibration data.  Defaults to -1, which means no limit on the number of the calibration data. |
| remove\_npy | bool (optional) | If True, remove pre-existing numpy files.  Defaults to False. |
| seed | int (optional) | Random seed.  Defaults to 2023. |
| save\_calib\_msg | bool (optional) | If True, save calibration data dictionary as MSGpack file.  Defaults to False. |
| msg\_path | string (optional) | Path to save MSGpack file  If not provided, it automatically generate the path with dataname and number of calibration data.  Defaults to None. |

Example codes for using these functions are provided in the [## Preparing Calibration Data](# Preparing Calibration Data) section.

## Pre-processing Configurations

qubee supports the following pre-processing functions to make calibration data.

Table 7‑12. Pre-processing function API

| Pre-processing Type | Description |
| --- | --- |
| GetImage | Get image tensor from image path using cv2 backend or image tensor.  Note that this should be at the top of the list. |
| Pad | Pad image tensor. |
| Normalize | Normalize image tensor. |
| ResizeTorch | Resize the input image to the given size using [torchvision.transforms.functional.resize](https://pytorch.org/vision/0.14/generated/torchvision.transforms.functional.resize.html?highlight=resize#torchvision.transforms.functional.resize) |
| Resize | Resize image tensor to the given size using [cv2.resize](https://docs.opencv.org/4.6.0/da/d54/group__imgproc__transform.html#ga47a974309e9102f5f08231edc7e7529d). |
| CenterCrop | Center crop the image tensor. |
| SetOrder | Set the order of axes of the given image tensor.  Note that this should be at the very end. |

You can write a yaml file as follows:

|  |
| --- |
| [Pre-processing Type]  [Parameter]: [Argument]  ... |

|  |
| --- |
| *# Example*  GetImage:  to\_float32: **false**  channel\_order: RGB  ResizeTorch:  size: [256, 256]  interpolation: blinear  CenterCrop:  size: [224, 224]  Normalize:  mean: [0.485, 0.456, 0.406]  std: [0.229, 0.224, 0.225]  to\_float: **true**  SetOrder:  shape: HWC |

### Pre-processing Parameters

Table 7‑13. GetImage

| Parameter | Type | Description |
| --- | --- | --- |
| to\_float32 | bool (optional) | If True, set dtype as float32.  Defaults to False. |
| channel\_order | string (optional) | Channel order to load. Upper cases will be converted into lower cases. Defaults to "bgr". |

Table 7‑14. Pad

| Parameter | Type | Description |
| --- | --- | --- |
| shape | Tuple[int] (optional) | Expected padding shape (h, w). Defaults to None. |
| size\_divisor | int (optional) | Pad images so that the the resulting image's width and height are divisible by size\_divisor.  Defaults to None. |
| pad\_val | float (optional) | Values to be filled in padding areas when padding\_mode is 'constant'.  Defaults to 0. |
| right\_bottom | bool (optional) | If True, it only pads to right and bottom. Defaults to False. |

Table 7‑15. Normalize

| Parameter | Type | Description |
| --- | --- | --- |
| mean | List[float] or np.ndarray | Normalization mean. |
| std | List[float] or np.ndarray | Normalization standard deviation. |
| to\_float | bool (optional) | Normalize image between [0, 255] into [0, 1] by dividing by 255 before normalizing with the mean and std.  Defaults to False. |

Table 7‑16. ResizeTorch

| Parameter | Type | Description |
| --- | --- | --- |
| size | List[int] | Desired output size, i.e., height and width. |
| interpolation | string | Interpolation method,  accepted values are "nearest", "bilinear", "bicubic", "box", "hamming", "lanczos". |

Table 7‑17. Resize

| Parameter | Type | Description |
| --- | --- | --- |
| img\_scale | float or Tuple[int, int] | The scaling factor or maximum size (h, w).  If it is a float number, then the image will be rescaled by this factor, else if it is a tuple of 2 integers, then the image will be rescaled as large as possible within the scale. |
| keep\_ratio | bool | Whether to keep the aspect ratio when resizing the image. Defaults to False. |
| interpolation | string | Interpolation method,  accepted values are "nearest", "bilinear", "bicubic", "area", "lanczos". |

Table 7‑18. CenterCrop

| Parameter | Type | Description |
| --- | --- | --- |
| size | List[int] | Desired output height and width. |

Table 7‑19. SetOrder

| Parameter | Type | Description |
| --- | --- | --- |
| shape | string | Desired data layout format, accepted values are "HWC", "CHW", "BHWC", "BCHW".  Defaults to "HWC". |

# Open Source License Notice

**PyTorch**

* <https://github.com/pytorch/pytorch>
* BSD-like License

**TensorFlow**

* <https://github.com/tensorflow/tensorflow>
* Apache 2.0 License

**ONNX**

* <https://github.com/onnx/onnx>
* Apache 2.0 License

**ONNX Runtime**

* <https://github.com/microsoft/onnxruntime>
* MIT License

**Keras**

* <https://github.com/keras-team/keras>
* Apache 2.0 License

# Copyright

Copyrightⓒ 2019-present, Mobilint, Inc. All rights reserved.