ESPRESSIF

ESP-DL User Guide

Release latest

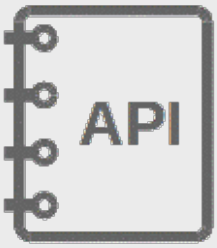
Espressif Systems

Apr 17, 2026

CONTENTS

Get Started

Tutorials



API Reference

CHAPTER

ONE

INTRODUCTION

1.1 Introduction

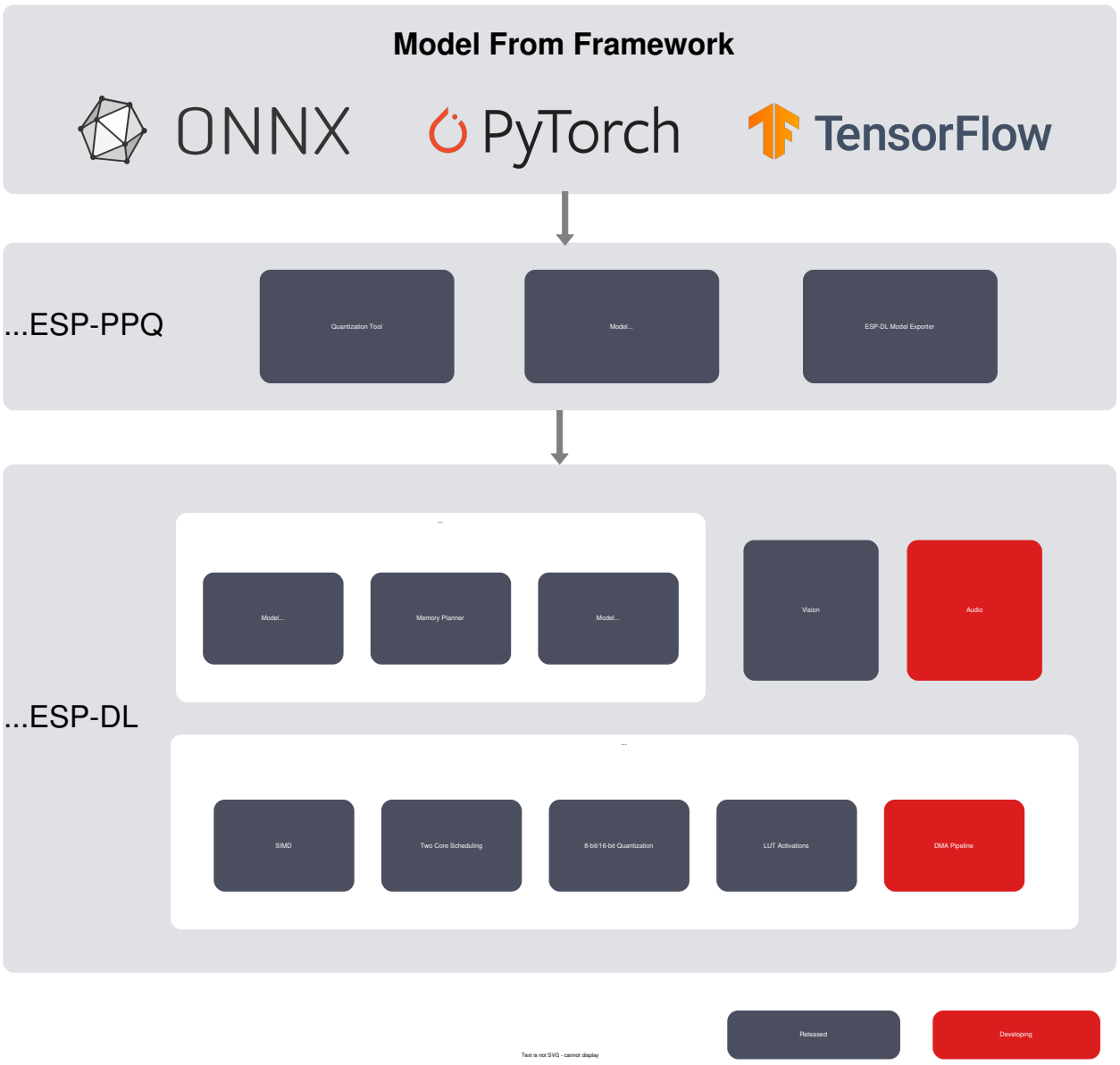
ESP-DL is a lightweight and efficient neural network inference framework designed specifically for ESP series chips. With ESP-DL, you can easily and quickly develop AI applications using Espressif's System on Chips (SoCs).

1.1.1 Overview

ESP-DL offers APIs to load, debug, and run AI models. The framework is easy to use and can be seamlessly integrated with other Espressif SDKs. ESP-PPQ serves as the quantization tool for ESP-DL, capable of quantizing models from ONNX, Pytorch, and TensorFlow, and exporting them into the ESP-DL standard model format.

- **ESP-DL Standard Model Format**: This format is similar to ONNX but uses FlatBuffers instead of Protobuf, making it more lightweight and supporting zero-copy deserialization, with a file extension of *.espdl*.
- **Efficient Operator Implementation**: ESP-DL efficiently implements common AI operators such as Conv, Gemm, Add, and Mul. The list of supported operators: operator_support_state.md
- **Static Memory Planner**: The memory planner automatically allocates different layers to the optimal memory location based on the user-specified internal RAM size, ensuring efficient overall running speed while minimizing memory usage.
- **Dual Core Scheduling**: Automatic dual-core scheduling allows computationally intensive operators to fully utilize the dual-core computing power. Currently, Conv2D and DepthwiseConv2D support dual-core scheduling.
- **8bit LUT Activation**: All activation functions except for ReLU and PReLU are implemented using an 8-bit LUT (Look Up Table) method in ESP-DL to accelerate inference. You can use any activation function, and their computational complexity remains the same.

The framework figures below illustrate the overall architecture of ESP-DL.



1.2 ESP-DL Project Organization

ESP-DL's modular design enables efficient development, maintenance, and scalability. The project is organized as follows:

1.2.1 dl (Deep Learning)

Core deep learning modules and tools, divided into submodules:

- **model** Loads, manages, and allocates memory for deep learning models. Includes `dl_model_base` and `dl_memory_manager`.
- **module** Interfaces for 60+ neural network operators (convolution, pooling, activation, etc.). Files: `dl_module_base.hpp`, `dl_module_conv.hpp`, `dl_module_pool.hpp`, `dl_module_relu.hpp`, etc.
- **base** Implements operations for chips (esp32, esp32s3, esp32p4) with ISA-specific assembly support. Includes operator implementations in `dl_base_conv2d.cpp/hpp`, `dl_base_avg_pool2d.cpp/hpp`, etc., and ISA-specific code in `isa/` subdirectories.
- **math** Mathematical operations (matrix functions). Files: `dl_math.hpp` and `dl_math_matrix.hpp`.
- **tool** Auxiliary functions (utility tools). Files: `dl_tool.hpp` and `dl_tool.cpp`. Includes ISA-specific tools in `isa/` subdirectories.
- **tensor** Tensor classes and operations. Files: `dl_tensor_base.hpp`.

1.2.2 vision

Computer vision modules divided into submodules:

- **classification** Image classification (model inference). Inference: `dl_cls_base`. Post-processors: `imagenet_cls_postprocessor`, `hand_gesture_cls_postprocessor`, `dl_cls_postprocessor`.
- **recognition** Feature extraction (model inference). Feature database management (Enroll, delete, query). Pre-processor: `dl_feat_image_preprocessor`. Inference: `dl_feat_base`. Post-processor: `dl_feat_postprocessor`. Database: `dl_recognition_database`
- **image** Image processing (resize, crop, warp affine). Color conversion (pixel, img). Image preprocessor (pipeline of resize, crop, color conversion, normalization, quantization). Image decoding/encoding (JPEG/BMP). Draw utility (point, hollow rectangle).

  Image process: `dl_image_process`. Color conversion: `dl_image_color`. Image preprocessor: `dl_image_preprocessor`. Image decoding/encoding: `dl_image_jpeg`, `dl_image_bmp`. Draw utility: `dl_image_draw`.
- **detect** Object detection (model inference). Inference: `dl_detect_base`. Post-processors: `dl_detect_yolo11_postprocessor`, `dl_detect_espdet_postprocessor`, `dl_detect_msr_postprocessor`, `dl_detect_mnp_postprocessor`, `dl_detect_pico_postprocessor`. Pose estimation: `dl_pose_yolo11_postprocessor`.

1.2.3 audio

Audio processing modules divided into submodules:

- **common** Common audio utilities. Files: `dl_audio_common.cpp/hpp`, `dl_audio_wav.cpp/hpp`.
- **speech_features** Speech feature extraction. Files: `dl_speech_features.cpp/hpp` (base class), `dl_fbank.cpp/hpp` (Filter Bank), `dl_mfcc.cpp/hpp` (MFCC), `dl_spectrogram.cpp/hpp` (Spectrogram).

1.2.4 fbs_loader (FlatBuffers Loader)

Handles FlatBuffers models:

- **include** Headers: `fbs_loader.hpp`, `fbs_model.hpp`.
- **src** Implementations: `fbs_loader.cpp`.
- **lib/** Pre-compiled libraries for different targets: `esp32/`, `esp32s3/`, `esp32p4/`.
- **espidl.fbs** FlatBuffers schema file.
- **pack_espdl_models.py** Model packing script.

1.2.5 Other Files

- **CMakeLists.txt** Project build configuration.
- **idf_component.yml** Component metadata (name, version, dependencies).
- **README.md** Project documentation and usage.
- **LICENSE** License terms.

CHAPTER

TWO

GETTING STARTED

2.1 Hardware Requirements

- An ESP32-S3 or ESP32-P4 development board. Recommended: ESP32-S3-EYE or ESP32-P4-Function-EV-Board
- PC (Linux)

Note:

- Some boards currently use Type C connectors. Make sure you use the right cable to connect the board!
- ESP-DL also supports ESP32, but its operator implementations are written in C, so the execution speed on ESP32 will be significantly slower than on ESP32-S3 or ESP32-P4. If needed, you can manually add compilation configuration files to your project—the function interface calls in ESP-DL remain identical. Note:
  - When quantizing **ESP32** platform models using **ESP-PPQ**, set the target to `c`.
  - When deploying **ESP32** platform models using **ESP-DL**, set the project compilation target to `esp32`.

2.2 Software Requirements

2.2.1 ESP-IDF

ESP-DL runs based on ESP-IDF. For detailed instructions on how to get ESP-IDF, see the ESP-IDF Programming Guide.

Note: Please use `release/v5.3` or higher version of ESP-IDF.

2.2.2 ESP-PPQ

ESP-PPQ is a quantization tool based on ppq, and its source code is fully open-sourced. ESP-PPQ adds Espressif's customized quantizer and exporter based on PPQ, which makes it convenient for users to select quantization rules that match ESP-DL according to different chip selections, and export them to standard model files that can be directly loaded by ESP-DL. ESP-PPQ is compatible with all PPQ APIs and quantization scripts. For more details, please refer to PPQ documents and videos. If you want to quantize your model, you can install esp-ppq using the following method:

Method 1: Install the package using pip

```
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/
↪cpu
pip install esp-ppq
```

Method 2: Install from source with pip to stay synchronized with the master branch

```
git clone https://github.com/espressif/esp-ppq.git
cd esp-ppq
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/
↪cpu
pip install -e .
```

Method 3: Install the package using uv

```
uv pip install "esp-ppq[cpu]" --torch-backend=cpu
# GPU
# uv pip install "esp-ppq[cpu]" --torch-backend=cu124
# AMD GPU
# uv pip install "esp-ppq[cpu]" --torch-backend=rocm6.2
# Intel XPU
# uv pip install "esp-ppq[cpu]" --torch-backend=xpu
```

Method 4: Install from source using uv to stay in sync with the master branch

```
git clone https://github.com/espressif/esp-ppq.git
cd esp-ppq
uv pip install torch torchvision torchaudio --index-url https://download.pytorch.org/
↪whl/cpu
uv pip install -e .
```

Method 5: Use esp-ppq with docker

```
docker build -t esp-ppq:your_tag https://github.com/espressif/esp-ppq.git
```

Note:

- The example code installs the Linux PyTorch CPU version. Please install the appropriate PyTorch version based on your actual needs.
- If installing the package with uv, simply modify the `--torch-backend` parameter, which will override the PyTorch URLs index configured in the project.

2.3 Quick Start

ESP-DL provides some out-of-the-box examples

2.3.1 Example Compile & Flash

```
idf.py set-target [Soc]
idf.py flash monitor -p [PORT]
```

Replace `[Soc]` with the specific chip, currently supports `esp32s3` and `esp32p4`. The example does not yet include the model and compilation configuration files for `esp32`.

2.3.2 Example Configuration

```
idf.py menuconfig
```

Some examples contain configurable options that can be configured using `idf.py menuconfig` after specifying the chip using `idf.py set-target`.

2.3.3 Trouble shooting

Check ESP-IDF doc

See ESP-IDF DOC

Erase FLASH & Clear Example

```
idf.py erase-flash -p [PORT]
```

Delete `build/`, `sdkconfig`, `dependencies.lock`, `managed_components/` and try again.

2.4 Model Quantization

First, please refer to operator_support_state.md to ensure that the operators in your model are supported.

ESP-DL must use the proprietary format `.espdl` for model deployment. Deep learning models need to be quantized and converted to the format before they can be used. ESP-PPQ provides two interfaces, `espdl_quantize_onnx` and `espdl_quantize_torch`, to support ONNX models and PyTorch models to be exported as `.espdl` models. Other deep learning frameworks, such as TensorfFlow, PaddlePaddle, etc., need to convert the model to ONNX first. So make sure your model can be converted to ONNX model. For more details, please refer to:

- *How to quantize model*
- *How to quantize MobileNetV2*
- *How to quantize YOLO11n*
- *How to quantize YOLO11n-pose*
- *How to quantize streaming model*

2.5 Model deployment

ESP-DL provides a series of APIs to quickly load and run models. For more details, see:

- *How to load & test & profile model*
- *How to run model*
- *How to deploy streaming model*

CHAPTER

THREE

TUTORIALS

3.1 How to quantize model

ESP-DL must use a proprietary format `.espdl` for model deployment. This is a quantized model format that supports 8bit and 16bit. In this tutorial, we will take quantize_sin_model as an example to show how to use ESP-PPQ to quantize and export a `.espdl` model. The quantization method is Post Training Quantization (PTQ).

3.1.1 Preparation

Install ESP_PPQ

3.1.2 Pre-trained model

```
python sin_model.py
```

Run sin_model.py . This script trains a simple Pytorch model to fit the sin function in the range [0, 2pi]. After training, the corresponding .pth weights will be saved and the ONNX model will be exported.

Note: ESP-PPQ provides two interfaces, `espdl_quantize_onnx` and `espdl_quantize_torch`, to support ONNX models and PyTorch models. Other deep learning frameworks, such as TensorfFlow, PaddlePaddle, etc., need to be converted to ONNX first.

- Convert TensorFlow to ONNX tf2onnx
- Convert TFLite to ONNX tflite2onnx
- Convert TFLite to TensorFlow tflite2tensorflow
- Convert PaddlePaddle to ONNX paddle2onnx

3.1.3 Quantize and export `.espdl`

Reference quantize_torch_model.py and quantize_onnx_model.py , learn how to use the `espdl_quantize_onnx` and `espdl_quantize_torch` interfaces to quantize and export the `.espdl` model.

After executing the script, three files will be exported:

- `**.espdl`: ESPDL model binary file, which can be directly used for chip reasoning.
- `**.info`: ESPDL model text file, used to debug and determine whether the `.espdl` model is exported correctly. Contains model structure, quantized model weights, test input/output and other information.
- `**.json`: Quantization information file, used to save and load quantization information.

Note:

1. The `.espdl` models of different platforms cannot be mixed, otherwise the inference results will be inaccurate.
   - The `ESP32` uses `ROUND_HALF_UP` as its rounding strategy.
     - When quantizing **ESP32** platform models using **ESP-PPQ**, set the target to `c`. Because ESP-DL implements its operators in C.
     - When deploying **ESP32** platform models using **ESP-DL**, set the project compilation target to `esp32`.
   - The ROUND strategy used by `ESP32S3` is `ROUND_HALF_UP`.
   - The ROUND strategy used by `ESP32P4` is `ROUND_HALF_EVEN`.
2. The quantization strategy currently used by ESP-DL is symmetric quantization + POWER OF TWO.

Add test input/output

To verify whether the inference results of the model on the board are correct, you first need to record a set of test input/output on the PC. By turning on the `export_test_values` option in the api, a set of test input/output can be saved in the `.espdl` model. One of the `input_shape` and `inputs` parameters must be specified. The `input_shape` parameter uses a random test input, while `inputs` can use a specific test input. The values of the test input/output can be viewed in the `.info` file. Search for `test inputs value` and `test outputs value` to view them.

Quantized model inference & accuracy evaluation

`espdl_quantize_onnx` and `espdl_quantize_torch` APIs will return `BaseGraph`. Use `BaseGraph` to build the corresponding `TorchExecutor` to use the quantized model for inference on the PC side.

```
executor = TorchExecutor(graph=quanted_graph, device=device)
output = executor(input)
```

The output obtained by quantized model inference can be used to calculate various accuracy metrics. Since the board-side `esp-dl` inference result can be aligned with `esp-ppq`, these metrics can be used directly to evaluate the accuracy of the quantized model.

Note:

1. Currently esp-dl only supports batch_size of 1, and does not support multi-batch or dynamic batch.
2. The test input/output and the quantized model weights in the `.info` file are all 16-byte aligned. If the length is less than 16 bytes, it will be padded with 0.

3.1.4 Advanced Quantization Methods

If you want to further improve the performance of the quantized model, please try the the following advanced quantization methods:

Post Training Quantization (PTQ)

- *Mixed precision quantization*
- *Layerwise equalization quantization*
- *Horizontal Layer Split Quantization*

Quantization Aware Training (QAT)

- *YOLO11n Quantization-Aware Training*
- *YOLO11n-pose Quantization-Aware Training*

3.2 How to load & test & profile model

In this tutorial, we will show you how to load, test, profile an espdl model. example

- *Test whether on-board model inference is correct*
- *Profile model memory usage*
- *Profile model inference latency*
- *Combined profiling: profile() method*

3.2.1 Preparation

1. *Install ESP_IDF*
2. *how_to_quantize_model*

3.2.2 Load model from `rodata`

This method embeds the model file directly into the application's `.rodata` section in FLASH. It's the simplest approach but has the drawback that the model gets re-flashed every time the application code changes.

1. **Add model file in** `CMakeLists.txt`

   To embed the `.espdl` model file into the `.rodata` section, add the following code to your `CMakeLists.txt`. The first few lines should be placed before `idf_component_register()` and the last line after `idf_component_register()`.

```
idf_build_get_property(component_targets __COMPONENT_TARGETS)
if ("___idf_espressif__esp-dl" IN_LIST component_targets)
   idf_component_get_property(espdl_dir espressif__esp-dl COMPONENT_DIR)
elseif("___idf_esp-dl" IN_LIST component_targets)
   idf_component_get_property(espdl_dir esp-dl COMPONENT_DIR)
endif()
set(cmake_dir ${espdl_dir}/fbs_loader/cmake)
include(${cmake_dir}/utilities.cmake)
set(embed_files your_model_path/model_name.espdl)

idf_component_register(...)

target_add_aligned_binary_data(${COMPONENT_LIB} ${embed_files} BINARY)
```

2. **Load the model in the program**

   Include the header file:

```
#include "dl_model_base.hpp"
```

   Declare the model symbol and create the model:

```
// The symbol name is composed of three parts: prefix "_binary_", filename "model_
↪espdl", and suffix "_start"
extern const uint8_t model_espdl[] asm("_binary_model_espdl_start");

// Basic usage - loads model with default parameters
dl::Model *model = new dl::Model((const char *)model_espdl, fbs::MODEL_LOCATION_
↪IN_FLASH_RODATA);

// Advanced usage with custom parameters:
```

(continues on next page)

(continued from previous page)

```
// - Keep parameters in FLASH (saves PSRAM/internal RAM, but lower performance)
// - Limit internal RAM usage to 0 bytes (use PSRAM first)
// - Use greedy memory manager
// - No encryption key
// - param_copy = false (keep parameters in FLASH)
// dl::Model *model = new dl::Model((const char *)model_espdl,
//                                  fbs::MODEL_LOCATION_IN_FLASH_RODATA,
//                                  0,  // max_internal_size
//                                  dl::MEMORY_MANAGER_GREEDY,
//                                  nullptr,  // key
//                                  false);   // param_copy
```

Note: Performance and Memory Trade-offs:

- **Flashing Time:** When using *Load model from rodata*, the model file is embedded in the application binary and gets re-flashed every time you modify your code. For large models, this increases flashing time. Consider *Load model from partition* or *Load model from sdcard* to avoid this.
- **Memory vs Performance:** The `param_copy` parameter controls whether model parameters are copied from FLASH to faster memory (PSRAM/internal RAM). Setting `param_copy=false` saves RAM but reduces inference performance since FLASH access is slower. Only disable parameter copying if RAM is extremely tight.
- **App Partition Size:** Large models embedded in `.rodata` may require increasing the app partition size in `partition.csv`.

3.2.3 Load model from `partition`

This method stores the model in a separate FLASH partition, allowing you to update the model independently of the application code.

1. **Add model information in** `partition.csv`

   Create or modify your `partition.csv` file to include a partition for the model. For details on partition tables, refer to the ESP-IDF partition table documentation.

   ```
   # Name, Type, SubType, Offset, Size, Flags
   factory, app, factory, 0x010000, 4000K,
   model, data, spiffs, , 4000K,
   ```

   - **Name:** Any meaningful name (max 16 characters including null terminator)
   - **Type:** `data`
   - **SubType:** `spiffs` (required for model storage)
   - **Offset:** Leave blank for automatic calculation
   - **Size:** Must be larger than the model file size

2. **Add model flashing information in** `CMakeLists.txt`

   ```
   idf_component_register(...)
   set(image_file your_model_path/model_name.espdl)
   esptool_py_flash_to_partition(flash "model" "${image_file}")
   ```

   The second parameter in `esptool_py_flash_to_partition` must match the `Name` field in `partition.csv`.

3. **Load the model in the program**

   Include the header file:

```
#include "dl_model_base.hpp"
```

   Create the model instance:

```
// Basic usage - loads model with default parameters
dl::Model *model = new dl::Model("model", fbs::MODEL_LOCATION_IN_FLASH_PARTITION);

// Advanced usage - keep parameters in FLASH to save RAM
// dl::Model *model = new dl::Model("model",
//                                  fbs::MODEL_LOCATION_IN_FLASH_PARTITION,
//                                  0,  // max_internal_size
//                                  dl::MEMORY_MANAGER_GREEDY,
//                                  nullptr,  // key
//                                  false);   // param_copy
```

   The first parameter (partition label) must match the `Name` field in `partition.csv`.

Note: **Flashing Optimization:** Use `idf.py app-flash` instead of `idf.py flash` to flash only the application partition without re-flashing the model partition. This significantly reduces flashing time during development.

3.2.4 Load model from `sdcard`

This method loads the model from an SD card, which is useful when FLASH space is limited or when you need to update models frequently without re-flashing.

1. **Prepare the SD card**
   - **Format:** The SD card should be formatted as FAT32. If not, it will be automatically formatted when mounted (data will be lost).
   - **Backup:** Always backup SD card data before using it with ESP-DL.
2. **Mount the SD card**
   - **Using BSP (Board Support Package):**

     Enable `CONFIG_BSP_SD_FORMAT_ON_MOUNT_FAIL` in menuconfig to allow automatic formatting.

```
#include "bsp/esp-bsp.h"
ESP_ERROR_CHECK(bsp_sdcard_mount());
```

   - **Without BSP:**

     Configure the mount options with `format_if_mount_failed = true`.

```
#include "esp_vfs_fat.h"
#include "sdmmc_cmd.h"

esp_vfs_fat_sdmmc_mount_config_t mount_config = {
    .format_if_mount_failed = true,
    .max_files = 5,
    .allocation_unit_size = 16 * 1024
};
// Mount SD card (implementation depends on your hardware)
```

3. **Copy model to SD card**

   Copy your `.espdl` model file to the SD card (e.g., to the root directory as `model.espdl`).

4. **Load the model in the program**

   Include the header file:

```
#include "dl_model_base.hpp"
```

   Create the model instance:

```
// Basic usage with BSP
ESP_ERROR_CHECK(bsp_sdcard_mount());
dl::Model *model = new dl::Model("/sdcard/model.espdl", fbs::MODEL_LOCATION_IN_
↪SDCARD);

// Or with custom path
// dl::Model *model = new dl::Model("/sdcard/models/my_model.espdl", fbs::MODEL_
↪LOCATION_IN_SDCARD);

// Don't forget to unmount when done
// ESP_ERROR_CHECK(bsp_sdcard_unmount());
```

   For non-BSP usage, mount the SD card first, then create the model similarly.

Note: **Performance Considerations:** Loading from SD card is slower than from FLASH because the model data must be copied from the SD card to RAM. However, this method saves FLASH space and allows easy model updates by swapping SD cards.

3.2.5 Test whether on-board model inference is correct

The `test()` method verifies that the model produces correct inference results by comparing them against ground truth values embedded in the model file.

Prerequisites:

- The `.espdl` model must be exported with **test inputs and outputs** enabled in ESP-PPQ (use the `export_test_values` option).
- For deployment, you can export a version without test data to reduce model size.

API: `esp_err_t dl::Model::test()`

Returns: `ESP_OK` if all tests pass, `ESP_FAIL` otherwise.

Usage:

```
#include "dl_model_base.hpp"

// After creating the model...
esp_err_t ret = model->test();
if (ret == ESP_OK) {
    ESP_LOGI(TAG, "Model test passed!");
} else {
    ESP_LOGE(TAG, "Model test failed!");
}
```

(continues on next page)

(continued from previous page)

```
// Or using the convenience macro:
ESP_ERROR_CHECK(model->test());
```

How it works:

1. Loads test input tensors embedded in the model
2. Runs inference through all model layers
3. Compares each output against the ground truth values (with tolerance for quantization errors)
4. Reports success or failure for each output

Note for INT16 models: Due to quantization rounding errors, INT16 models allow ±1 difference in comparison.

3.2.6 Profile model memory usage

The `profile_memory()` method prints a detailed breakdown of memory usage across different memory types (internal RAM, PSRAM, FLASH).

API: `void dl::Model::profile_memory()`

Usage:

```
#include "dl_model_base.hpp"

// After creating and testing the model...
model->profile_memory();
```

Output includes:

Name	Explanation
`fbs_model` `parameter`	FlatBuffers model structure (includes model metadata, graph structure, tensor shapes, etc.) Model parameters stored within the FlatBuffers model (sub-item of fbs_model)
`parameter_copy`	Parameters copied from FLASH to faster memory (PSRAM/internal RAM). Only present when `param_copy=true` (default). Improves inference performance.
`variable`	Memory allocated for model inputs, outputs, and intermediate tensors by the memory manager.
`others`	Miscellaneous memory usage (class member variables, alignment overhead, etc.). Usually very small.
`total`	Total memory usage across all categories.

Memory types shown: Internal RAM, PSRAM, and FLASH usage for each category.

3.2.7 Profile model inference latency

The `profile_module()` method prints detailed latency information for each module (layer) in the model.

API: `void dl::Model::profile_module(bool sort_module_by_latency = false)`

Parameters: - `sort_module_by_latency`: If `true`, modules are sorted by latency (highest first). If `false` (default), modules are shown in ONNX topological order.

Usage:

```
// Default: topological order
model->profile_module();

// Sorted by latency (highest first)
model->profile_module(true);
```

Output includes: - Module name - Module type (operation type) - Inference latency in microseconds (or cycles if `DL_LOG_LATENCY_UNIT` is enabled) - Total inference latency at the end

3.2.8 Combined profiling: profile() method

The `profile()` method combines `profile_memory()` and `profile_module()` for comprehensive analysis.

API: `void dl::Model::profile(bool sort_module_by_latency = false)`

Usage:

```
// Comprehensive profiling in topological order
model->profile();

// Comprehensive profiling sorted by latency
model->profile(true);
```

This is the most convenient way to get both memory and performance analysis in one call.

3.3 How to run model

In this tutorial, we will introduce the most basic model inference process. example

- *Dequantize* `dl::TensorBase`
- *Model Inference*

3.3.1 Preparation

Install ESP_IDF

3.3.2 Load model

How to load model

3.3.3 Get model input/output.

```
std::map<std::string, dl::TensorBase *> model_inputs = model->get_inputs();
dl::TensorBase *model_input = model_inputs.begin()->second;
std::map<std::string, dl::TensorBase *> model_outputs = model->get_outputs();
dl::TensorBase *model_output = model_outputs.begin()->second;
```

You can get the input/output names and the corresponding `dl::TensorBase` with `get_inputs()` and `get_outputs()` api. For more information, see *dl::TensorBase documentation*.

Note: ESP-DL's memory manager allocates a whole block of memory for each model's input/intermediate result/output. Since they share this memory, when the model is inferencing, the later results will overwrite the previous results. In other words, the data in `model_input` may be overwritten by `model_output` or other intermediate results after the model inference is completed.

3.3.4 Quantize Input

8-bit and 16-bit quantized models accept inputs of type `int8_t` and `int16_t` respectively. `float` inputs must be quantized to the one of them according to `exponent` before being fed into the model. Calculation formula:

$$Q = \text{Clip}\left(\text{Round}\left(\frac{R}{\text{Scale}}\right), \text{MIN}, \text{MAX}\right)$$

$$\text{Scale} = 2^{\text{Exp}}$$

Where:

- R is the floating point number to be quantized.
- Q is the integer value after quantization, which needs to be clipped within the range [MIN, MAX].
- MIN is the minimum integer value, when 8bit, MIN = -128, when 16bit, MIN = -32768.
- MAX is the maximum integer value, when 8bit, MAX = 127, when 16bit, MAX = 32767.

Quantize a single value

```
float input_v = VALUE;
// Note that dl::quantize accepts inverse of scale as the second input, so we use DL_
↪RESCALE here.
int8_t quant_input_v = dl::quantize<int8_t>(input_v, DL_RESCALE(model_input->
↪exponent));
```

Quantize `dl::TensorBase`

```
// assume that input_tensor already contains the float input data.
dl::TensorBase *input_tensor;
model_input->assign(input_tensor);
```

3.3.5 Dequantize output

8bit and 16bit quantized model, get `int8_t` and `int16_t` type output respectively. Must be dequantized according to `exponent` to get floating point output. Calculation formula:

$$R' = Q \times \text{Scale}$$

$$\text{Scale} = 2^{\text{Exp}}$$

Where:

- R' is the approximate floating point value recovered after dequantization.
- Q is the integer value after quantization.

Dequantize a single value

```
int8_t quant_output_v = VALUE;
float output_v = dl::dequantize(quant_output_v, DL_SCALE(model_output->exponent));
```

Dequantize `dl::TensorBase`

```
// create a TensorBase filled with 0 of shape [1, 1]
dl::TensorBase *output_tensor = new dl::TensorBase({1, 1}, nullptr, 0, dl::DATA_TYPE_
↪FLOAT);
output_tensor->assign(model_output);
```

3.3.6 Model Inference

See:

- example
- *void dl::Model::run(runtime_mode_t mode)*
- *void dl::Model::run(TensorBase *input, runtime_mode_t mode)*
- *void dl::Model::run(std::map<std::string, TensorBase*> &user_inputs, runtime_mode_t mode, std::map<std::string, TensorBase*> user_outputs)*

3.4 Creating a New Module (Operator)

This tutorial guides you through the process of creating a new module in the `dl::module` namespace. The `Module` class serves as the base class for all modules, and you can extend this base class to create your custom module.

Note: The interface of modules in ESP-DL should be aligned with ONNX.

3.4.1 Understand the Base `Module` Class

The base class provides several virtual methods that must be overridden in your derived class.

- **Methods:**
  - *dl::module::Module::Module()*: Constructor to initialize the module.
  - *dl::module::Module::~Module()*: Destructor to release resources.
  - *dl::module::Module::get_output_shape()*: Calculates the output shape based on the input shape.
  - *dl::module::Module::forward()*: Runs the module, high-level interface.
  - *dl::module::Module::forward_args()*: Runs the module, low-level interface.
  - *dl::module::Module::deserialize()*: Creates a module instance from serialized information.
  - *dl::module::Module::print()*: Prints module information.

For more information, please refer to Module Class Reference.

3.4.2 Create a New Module Class

To create a new module, you need to derive a new class from the `Module` base class and override the necessary methods.

Example: Creating a `MyCustomModule` Class

For more examples, please refer to esp-dl/dl/module.

```
#include "module.h" // Include the header file where the Module class is defined

namespace dl {
namespace module {

class MyCustomModule : public Module {
public:
    // Constructor
    MyCustomModule(const char *name = "MyCustomModule",
                module_inplace_t inplace = MODULE_NON_INPLACE,
                quant_type_t quant_type = QUANT_TYPE_NONE)
        : Module(name, inplace, quant_type) {}

   // Destructor
    virtual ~MyCustomModule() {}

    // Override the get_output_shape method
    std::vector<std::vector<int>> get_output_shape(std::vector<std::vector<int>> &
↪input_shapes) override {
        // Implement the logic to calculate the output shape based on input shapes
        std::vector<std::vector<int>> output_shapes;
        // Example: Assume the output shape is the same as the input shape
        output_shapes.push_back(input_shapes[0]);
        return output_shapes;
    }

    // Override the forward method
    void forward(std::vector<dl::TensorBase *> &tensors, runtime_mode_t mode =␣
↪RUNTIME_MODE_AUTO) override {
       // Implement the logic to run the module
        // Example: Perform some operation on the tensors
        for (auto &tensor : tensors) {
           // Perform some operation on each tensor
        }
    }

    // Override the forward_args method
    void forward_args(void *args) override {
        // Implement the low-level interface logic
        // Example: Perform some operation based on the arguments
    }

    // Deserialize module instance by serialization information
    static Module *deserialize(fbs::FbsModel *fbs_model, std::string node_name){
        // Implement the logic to deserialize the module instance
        // The interface should be align with ONNX
    }

    // Override the print method
    void print() override {
        // Print module information
        ESP_LOGI("MyCustomModule", "Module Name: %s, Quant type: %d", name.c_str(),␣
↪quant_type);
    }
```

(continues on next page)

(continued from previous page)

```
};

} // namespace module
} // namespace dl
```

Register `MyCustomModule` Class

Once you have implemented `MyCustomModule` Class, register your module in dl_module_creator as a globally available module.

```
void register_dl_modules()
{
    if (creators.empty()) {
        ...
        this->register_module("MyCustomModule", MyCustomModule::deserialize);
    }
}
```

3.5 Implement Operators Automatically with AI Agent

This document describes how to install and use the `espdl-operator` skill in various Coding Agent tools (such as Claude, Cursor, OpenCode, etc.) for automated neural network operator implementation in the ESP-DL framework. **The following instructions use Linux environment as an example.**

3.5.1 What is `espdl-operator` skill

`espdl-operator` is an automated development skill for Coding Agents, used to implement, test, and optimize neural network operators in the ESP-DL framework. When you make operator-related requests to a Coding Agent (such as Claude, Cursor, OpenCode, etc.), this skill guides the AI to automatically complete the following tasks:

The Coding Agent will automatically:

1. **Analyze operator requirements** - Parse ONNX operator specifications, determine operator type and data type support
2. **Generate esp-dl C++ code** - Automatically create Module layer and Base layer header/implementation files

3. **Modify esp-ppq quantization config** - Register operator support in the quantization tool, configure layout patterns
4. **Create test cases** - Generate PyTorch/ONNX test models, configure test parameters
5. **Execute build and testing** - Run Docker builds, generate test data, execute hardware tests
6. **Verify result alignment** - Ensure inference results are consistent between esp-dl and esp-ppq

Core Value of the Skill:

- **End-to-end automation** - From requirements to runnable code, the Coding Agent automatically completes all steps
- **Cross-repository coordination** - Simultaneously modifies both esp-dl (C++) and esp-ppq (Python) codebases
- **Follows best practices** - Automatically applies ESP-DL code standards, directory structure, and testing workflows
- **Incremental development** - Supports new operator implementation, data type extension, and other scenarios

Applicable Scenarios:

Scenario	Example Request
Implement new operator	"Implement HardSwish operator with int8 and float32 support"
Add data type support	"Add int16 support for Tanh operator"
Quantization support	"Add quantization support for Mod"
Result alignment	"Verify if LogSoftmax results are consistent between esp-dl and esp-ppq"

3.5.2 Dependencies

Before using the `espdl-operator` skill, you need to install the following dependencies in advance:

Required Pre-installed Dependencies

Dependency	Purpose	Installation Command
Docker	For build and test environment	Official Installation Guide
uv	Python package manager	`curl -LsSf https://astral.sh/uv/install.sh | sh`
Git	Version control	`apt install git` (Ubuntu/Debian)

Dependencies Handled Automatically by Skill

The following dependencies **do not need manual installation**; the `espdl-operator` skill will handle them automatically during execution:

- **esp-ppq**: Python quantization toolkit - skill automatically installs from source in Docker container
- **Documentation generation scripts**: `gen_ops_markdown.py` and other tools - skill runs automatically
- **Docker image**: `espdl/idf-ppq` image - skill builds automatically (if not exists)
- **ESP-IDF**: Development framework - included in Docker image

Verify Dependency Installation

After installation, verify that the following commands work:

```
# Check Docker
docker --version

# Check uv
uv --version

# Check Git
git --version
```

If all commands output version information normally, the environment is ready to use the skill.

3.5.3 Installing/Placing Skill

Project Structure

First, confirm your project directory structure as follows (`esp_dl_project_1` is the project root directory, the root directory name can be arbitrary). Using opencode as an example, the full directory structure is shown below:

```
esp_dl_project_1/                    <-- Project root directory (all commands␣
↪executed here)
├── esp-dl/                          # ESP-DL main codebase
│   ├── esp-dl/                      # Core library source code (dl/, vision/, audio/,
↪ etc.)
│   ├── examples/                    # Example programs
│   ├── test_apps/                   # Test applications
│   ├── tools/                       # Tool scripts
│   └── ...
├── esp-ppq/                         # Quantization tool (same level as esp-dl)
│   ├── esp_ppq/                     # Main package source code
│   ├── pyproject.toml               # Project configuration file
│   └── ...
└── .opencode/                       # OpenCode configuration (needs to be created)
    └── skills/espdl-operator/       # skill installation location (points to esp-dl/
↪tools/agents/skills/espdl-operator/)
```

Note: The skill source code is located at `esp-dl/tools/agents/skills/espdl-operator/`. You need to copy or link it to `.opencode/skills/espdl-operator/` (or the corresponding directory for other Agents). The skill files include `SKILL.md` (main file) and `references/` (reference templates and checklists).

Important: Command Execution Location

All commands below must be executed in the project root directory ``esp_dl_project_1/``.

If you are unsure of your current location, first execute:

```
# Check current directory
pwd

# Should output something like: /home/username/workspace/esp_dl_project_1
# or /path/to/esp_dl_project_1

```

(continues on next page)

(continued from previous page)

```
# If not in project root, navigate there first
cd /path/to/esp_dl_project_1
```

Method 1: Use npx (Recommended - Simplest)

The easiest way to install the skill for OpenCode, Cursor, Claude Code, and other compatible tools is using npx:

```
npx skills add espressif/esp-dl --skill espdl-operator
```

Note: npx is the package runner that comes with Node.js (via npm). If you haven't installed Node.js yet, please refer to Node.js Installation Guide to install it first.

After running the command, the skill will be automatically installed and ready to use in your Coding Agent tool.

Method 2: Manual Installation

If you prefer manual installation or your tool doesn't support npx, follow the instructions below for your specific tool:

Note: The skill installation directory varies by Agent tool. Please choose the appropriate path based on the tool you are using.

OpenCode

Method 1: Copy Files

```
# Ensure you are in the project root directory esp_dl_project_1/
cd /path/to/esp_dl_project_1

# Create .opencode/skills directory
mkdir -p .opencode/skills/espdl-operator

# Copy from esp-dl/tools/agents/skills/espdl-operator to .opencode/skills/espdl-
↪operator
cp -r esp-dl/tools/agents/skills/espdl-operator/* .opencode/skills/espdl-operator/
```

Method 2: Use Symbolic Link (Recommended for Development, Keeps in Sync)

```
# Ensure you are in the project root directory esp_dl_project_1/
cd /path/to/esp_dl_project_1

# Create .opencode/skills directory
mkdir -p .opencode/skills

# Create symbolic link (using relative path)
# Note: from .opencode/skills/espdl-operator pointing to esp-dl/tools/agents/skills/
↪espdl-operator
ln -s ../../esp-dl/tools/agents/skills/espdl-operator .opencode/skills/espdl-operator

# Verify link is successful
```

(continues on next page)

(continued from previous page)

```
ls -la .opencode/skills/espdl-operator
# Should show SKILL.md and references/ directory
```

After starting OpenCode, the system will automatically load this skill.

Cursor

Method 1: Copy Files

```
# Ensure you are in the project root directory esp_dl_project_1/
cd /path/to/esp_dl_project_1

# Create Cursor skills directory
mkdir -p .cursor/skills/espdl-operator

# Copy skill files
cp -r esp-dl/tools/agents/skills/espdl-operator/* .cursor/skills/espdl-operator/
```

Method 2: Use Symbolic Link

```
# Ensure you are in the project root directory esp_dl_project_1/
cd /path/to/esp_dl_project_1

# Create .cursor/skills directory
mkdir -p .cursor/skills

# Create symbolic link
ln -s ../../esp-dl/tools/agents/skills/espdl-operator .cursor/skills/espdl-operator
```

Claude Desktop (Claude Code)

Method 1: Copy Files

```
# Ensure you are in the project root directory esp_dl_project_1/
cd /path/to/esp_dl_project_1

# Create Claude skills directory
mkdir -p .claude/skills/espdl-operator

# Copy skill files
cp -r esp-dl/tools/agents/skills/espdl-operator/* .claude/skills/espdl-operator/
```

Method 2: Use Symbolic Link

```
# Ensure you are in the project root directory esp_dl_project_1/
cd /path/to/esp_dl_project_1

# Create .claude/skills directory
mkdir -p .claude/skills

# Create symbolic link
ln -s ../../esp-dl/tools/agents/skills/espdl-operator .claude/skills/espdl-operator
```

3.5.4 Quick Start Example

Suppose you want to implement a new operator `MyOp`:

1. **Ensure skill is installed**

```
ls -la .opencode/skills/espdl-operator/SKILL.md
```

2. **Ask in Agent**

```
"Help me implement a MyOp operator with int8, int16, and float32 support"
```

3. **Agent will automatically**
   - Load the skill
   - Guide the Coding Agent through 9 phases
   - Generate necessary code files
   - Run Docker tests

3.5.5 Skill Trigger Usage

After installation, you can trigger the `espdl-operator` skill in the following ways:

Natural Language Trigger

Use the following keywords directly in conversation:

Chinese Trigger	English Trigger
"□□□□"	"implement operator"
"□□□□"	"add operator"
"□□□□"	"quantization support"
"□□□□"	"operator alignment"
"□□□□□□"	"add a new op"

Example Conversations

```
User: "Help me implement a Mod operator"
Agent: [Automatically loads espdl-operator skill and starts guiding]

User: "Add LogSoftmax operator to esp-dl"
Agent: [Automatically loads skill and provides implementation steps]
```

Explicit Invocation

If automatic triggering doesn't work, you can explicitly ask the Agent to use this skill:

```
"Use espdl-operator skill to help me implement Softmax operator"
"Following espdl-operator skill guidance, add int16 support for LogSoftmax"
```

3.5.6 Main Functionality Workflow

This skill guides the Coding Agent through the following main phases:

Phase 1: Research and Classification

- Understand ONNX operator specifications
- Determine operator type (Elementwise, Convolution, Pooling, etc.)
- Determine supported data types (int8, int16, float32)

Phase 2: Implement esp-dl Module Layer

- Create operator module header file (`dl_module_<op>.hpp`)
- Implement `get_output_shape()` and `forward()` methods
- Register operator in `dl_module_creator.hpp`

Phase 3: Implement esp-dl Base Layer

- Create C reference implementation (`dl_base_<op>.hpp/cpp`)

Phase 4: esp-ppq Integration

- Register quantization support in `EspdlQuantizer.py`
- Configure layout pattern in `espdl_typedef.py`

Phase 5: Configure Test Cases

- Add PyTorch/ONNX test model builders
- Configure test parameters in `op_cfg.toml`

Phase 6: Docker Build and Test

- Generate test cases (int8, int16, float32)
- Build test applications
- Run tests on hardware

Phase 7: SIMD Optimization (Optional)

- This part is not yet supported and will be iteratively improved in the future

Phase 8: Operator Alignment Verification

- Ensure esp-dl and esp-ppq inference results are consistent

Phase 9: Update Documentation

- Run `gen_ops_markdown.py` to update operator support status documentation

3.5.7 Troubleshooting

Skill Not Triggered

- Confirm skill directory is in the correct location (using opencode as example: `.opencode/skills/espdl-operator`)
- Try using explicit trigger words: "Use espdl-operator skill…"
- Confirm the Agent tool supports skills

Skill Workflow Not Fully Executed

- If some steps were not executed, explicitly invoke the `espdl-operator` skill, for example:

```
# If Docker commands for hardware flashing/testing were not executed, issue the␣
↪command in conversation:
Based on espdl-operator skill instructions, perform hardware flashing and testing,
↪ hardware is connected
```

Operator Implementation Quality Not Ideal

The role of this skill is to guide the AI to automatically complete operator code writing. Therefore, the final result is affected by two factors:

1. **The AI tool you use** (such as OpenCode, Cursor, Claude, etc.)
2. **The AI model's own coding capabilities** (different models have varying coding skills, comprehension abilities, and tool calling capabilities)

Based on our testing experience, the following combinations work well when implementing C language version operators (due to limited resources, many combinations have not been covered; you can try them yourself):

Coding Agent Tool	Model Used
Cursor	Claude Opus 4.6
OpenCode	Kimi 2.5 + GLM 5 or Claude Opus 4.6

If the generated code quality is poor, you can try:

- Switch to a more powerful AI model (such as Claude Opus, GPT-4, etc.)
- Try a Coding Agent tool with better results
- Manually implement following the detailed guidance in SKILL.md

Docker Issues

```
# Check if Docker is running
docker ps

# Rebuild image
cd esp-dl/tools/agents/skills/espdl-operator/assets/docker
docker build -t espdl/idf-ppq:latest .
```

Permission Issues

```
# Ensure device access permissions (Linux)
sudo usermod -a -G dialout $USER
# Log out and back in for changes to take effect
```

3.5.8 Related Resources

- **SKILL.md**: `esp-dl/tools/agents/skills/espdl-operator/SKILL.md` - Complete development guide
- **Templates**: `esp-dl/tools/agents/skills/espdl-operator/references/esp-dl-templates.md`
- **Checklist**: `esp-dl/tools/agents/skills/espdl-operator/references/esp-ppq-checklist.md`
- **esp-dl**: `esp-dl/` - esp-dl main codebase
- **esp-ppq**: `esp-ppq/` - esp-ppq quantization tool (same level directory as esp-dl)

3.6 How to deploy MobileNetV2

In this tutorial, we will introduce how to quantize a pre-trained MobileNetV2 model using ESP-PPQ and deploy the quantized MobileNetV2 model using ESP-DL.

3.6.1 Preparation

1. *Install ESP_IDF*
2. *Install ESP_PPQ*

3.6.2 Model quantization

Quantization script

Pre-trained model

Load the pre-trained model of MobileNet_v2 from torchvision. You can also download it from ONNX models or TensorFlow models:

```
import torchvision
from torchvision.models.mobilenetv2 import MobileNet_V2_Weights

model = torchvision.models.mobilenet.mobilenet_v2(weights=MobileNet_V2_Weights.
↪IMAGENET1K_V1)
```

Calibration dataset

The calibration dataset needs to be consistent with your model input format. The calibration dataset needs to cover all possible situations of your model input as much as possible to better quantize the model. Here we take the ImageNet dataset as an example to demonstrate how to prepare the calibration dataset.

Use torchvision to load the ImageNet dataset:

```
import torchvision.datasets as datasets
from torch.utils.data.dataset import Subset
dataset = datasets.ImageFolder(
   CALIB_DIR,
   transforms.Compose(
         [
            transforms.Resize(256),
            transforms.CenterCrop(224),
            transforms.ToTensor(),
            transforms.Normalize(
               mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]
            ),
         ]
   ),
)
dataset = Subset(dataset, indices=[_ for _ in range(0, 1024)])
dataloader = DataLoader(
   dataset=dataset,
   batch_size=BATCH_SIZE,
   shuffle=False,
   num_workers=4,
   pin_memory=False,
   collate_fn=collate_fn1,
)
```

8bit default configuration quantization

Quantization settings

```
target="esp32p4"
num_of_bits=8
batch_size=32
quant_setting = QuantizationSettingFactory.espdl_setting() # default setting
```

Quantization results

```
Analysing Graphwise Quantization Error::
Layer                                          | NOISE:SIGNAL POWER RATIO
/features/features.16/conv/conv.2/Conv:        | ████████████████████ | 48.831%
/features/features.15/conv/conv.2/Conv:        | ███████████████████  | 45.268%
/features/features.17/conv/conv.2/Conv:        | ██████████████████   | 43.112%
/features/features.18/features.18.0/Conv:      | ████████████████     | 41.586%
/features/features.14/conv/conv.2/Conv:        | ████████████████     | 41.135%
/features/features.13/conv/conv.2/Conv:        | ████████████         | 35.090%
/features/features.17/conv/conv.0/conv.0.0/Conv: | ███████████        | 32.895%
/features/features.16/conv/conv.1/conv.1.0/Conv: | ██████████         | 29.226%
/features/features.12/conv/conv.2/Conv:        | ██████████         | 28.895%
/features/features.16/conv/conv.0/conv.0.0/Conv: | █████████          | 27.808%
```

(continues on next page)

(continued from previous page)

```
/features/features.7/conv/conv.2/Conv:           | ██████████           | 27.675%
/features/features.10/conv/conv.2/Conv:          | ██████████           | 26.292%
/features/features.11/conv/conv.2/Conv:          | ██████████           | 26.085%
/features/features.6/conv/conv.2/Conv:           | ██████████           | 25.892%
/classifier/classifier.1/Gemm:                   | █████████            | 25.591%
/features/features.15/conv/conv.0/conv.0.0/Conv: | █████████            | 25.323%
/features/features.4/conv/conv.2/Conv:           | █████████            | 24.787%
/features/features.15/conv/conv.1/conv.1.0/Conv: | █████████            | 24.354%
/features/features.14/conv/conv.1/conv.1.0/Conv: | ████████             | 20.207%
/features/features.9/conv/conv.2/Conv:           | ████████             | 19.808%
/features/features.14/conv/conv.0/conv.0.0/Conv: | ████████             | 18.465%
/features/features.5/conv/conv.2/Conv:           | ███████              | 17.868%
/features/features.12/conv/conv.1/conv.1.0/Conv: | ███████              | 16.589%
/features/features.13/conv/conv.1/conv.1.0/Conv: | ███████              | 16.143%
/features/features.11/conv/conv.1/conv.1.0/Conv: | ██████               | 15.382%
/features/features.3/conv/conv.2/Conv:           | ██████               | 15.105%
/features/features.13/conv/conv.0/conv.0.0/Conv: | ██████               | 15.029%
/features/features.10/conv/conv.1/conv.1.0/Conv: | ██████               | 14.875%
/features/features.2/conv/conv.2/Conv:           | ██████               | 14.869%
/features/features.11/conv/conv.0/conv.0.0/Conv: | ██████               | 14.552%
/features/features.9/conv/conv.1/conv.1.0/Conv:  | ██████               | 14.050%
/features/features.8/conv/conv.1/conv.1.0/Conv:  | ██████               | 13.929%
/features/features.8/conv/conv.2/Conv:           | ██████               | 13.833%
/features/features.12/conv/conv.0/conv.0.0/Conv: | ██████               | 13.684%
/features/features.7/conv/conv.0/conv.0.0/Conv:  | █████                | 12.942%
/features/features.6/conv/conv.1/conv.1.0/Conv:  | █████                | 12.765%
/features/features.10/conv/conv.0/conv.0.0/Conv: | █████                | 12.251%
/features/features.5/conv/conv.1/conv.1.0/Conv:  | █████                | 11.186%
/features/features.17/conv/conv.1/conv.1.0/Conv: | ████                 | 11.070%
/features/features.9/conv/conv.0/conv.0.0/Conv:  | ████                 | 10.371%
/features/features.4/conv/conv.1/conv.1.0/Conv:  | ████                 | 10.356%
/features/features.6/conv/conv.0/conv.0.0/Conv:  | ████                 | 10.149%
/features/features.4/conv/conv.0/conv.0.0/Conv:  | ████                 | 9.472%
/features/features.8/conv/conv.0/conv.0.0/Conv:  | ████                 | 9.232%
/features/features.3/conv/conv.1/conv.1.0/Conv:  | ████                 | 9.187%
/features/features.1/conv/conv.1/Conv:           | ████                 | 8.770%
/features/features.5/conv/conv.0/conv.0.0/Conv:  | ███                  | 8.408%
/features/features.7/conv/conv.1/conv.1.0/Conv:  | ███                  | 8.151%
/features/features.2/conv/conv.1/conv.1.0/Conv:  | ███                  | 7.156%
/features/features.3/conv/conv.0/conv.0.0/Conv:  | ███                  | 6.328%
/features/features.2/conv/conv.0/conv.0.0/Conv:  | ██                   | 5.392%
/features/features.1/conv/conv.0/conv.0.0/Conv:  |                      | 0.875%
/features/features.0/features.0.0/Conv:          |                      | 0.119%
Analysing Layerwise quantization error:: 100
 ↪%|████████████████████████████████████████████████████████████████████████████████████████████████████████████
 ↪53/53 [08:44<00:00,  9.91s/it]
Layer                                            | NOISE:SIGNAL POWER RATIO
/features/features.1/conv/conv.0/conv.0.0/Conv:  | ████████████████████ | 14.303%
/features/features.0/features.0.0/Conv:          | ▉                    | 0.844%
/features/features.1/conv/conv.1/Conv:           | ▉                    | 0.667%
/features/features.2/conv/conv.1/conv.1.0/Conv:  | ▉                    | 0.574%
/features/features.3/conv/conv.1/conv.1.0/Conv:  | ▉                    | 0.419%
/features/features.15/conv/conv.1/conv.1.0/Conv: |                      | 0.272%
/features/features.9/conv/conv.1/conv.1.0/Conv:  |                      | 0.238%
/features/features.17/conv/conv.1/conv.1.0/Conv: |                      | 0.214%
/features/features.4/conv/conv.1/conv.1.0/Conv:  |                      | 0.180%
/features/features.11/conv/conv.1/conv.1.0/Conv: |                      | 0.151%
```

(continues on next page)

(continued from previous page)

```
/features/features.12/conv/conv.1/conv.1.0/Conv: |                    | 0.148%
/features/features.16/conv/conv.1/conv.1.0/Conv: |                    | 0.146%
/features/features.14/conv/conv.2/Conv:          |                    | 0.136%
/features/features.13/conv/conv.1/conv.1.0/Conv: |                    | 0.105%
/features/features.6/conv/conv.1/conv.1.0/Conv:  |                    | 0.105%
/features/features.8/conv/conv.1/conv.1.0/Conv:  |                    | 0.083%
/features/features.7/conv/conv.2/Conv:           |                    | 0.076%
/features/features.5/conv/conv.1/conv.1.0/Conv:  |                    | 0.076%
/features/features.3/conv/conv.2/Conv:           |                    | 0.075%
/features/features.16/conv/conv.2/Conv:          |                    | 0.074%
/features/features.13/conv/conv.0/conv.0.0/Conv: |                    | 0.072%
/features/features.15/conv/conv.2/Conv:          |                    | 0.066%
/features/features.4/conv/conv.2/Conv:           |                    | 0.065%
/features/features.11/conv/conv.2/Conv:          |                    | 0.063%
/classifier/classifier.1/Gemm:                   |                    | 0.063%
/features/features.2/conv/conv.0/conv.0.0/Conv:  |                    | 0.054%
/features/features.13/conv/conv.2/Conv:          |                    | 0.050%
/features/features.10/conv/conv.1/conv.1.0/Conv: |                    | 0.042%
/features/features.17/conv/conv.0/conv.0.0/Conv: |                    | 0.040%
/features/features.2/conv/conv.2/Conv:           |                    | 0.038%
/features/features.4/conv/conv.0/conv.0.0/Conv:  |                    | 0.034%
/features/features.17/conv/conv.2/Conv:          |                    | 0.030%
/features/features.14/conv/conv.0/conv.0.0/Conv: |                    | 0.025%
/features/features.16/conv/conv.0/conv.0.0/Conv: |                    | 0.024%
/features/features.10/conv/conv.2/Conv:          |                    | 0.022%
/features/features.11/conv/conv.0/conv.0.0/Conv: |                    | 0.021%
/features/features.9/conv/conv.2/Conv:           |                    | 0.021%
/features/features.14/conv/conv.1/conv.1.0/Conv: |                    | 0.020%
/features/features.7/conv/conv.1/conv.1.0/Conv:  |                    | 0.020%
/features/features.5/conv/conv.2/Conv:           |                    | 0.019%
/features/features.8/conv/conv.2/Conv:           |                    | 0.018%
/features/features.12/conv/conv.2/Conv:          |                    | 0.017%
/features/features.6/conv/conv.2/Conv:           |                    | 0.014%
/features/features.7/conv/conv.0/conv.0.0/Conv:  |                    | 0.014%
/features/features.3/conv/conv.0/conv.0.0/Conv:  |                    | 0.013%
/features/features.12/conv/conv.0/conv.0.0/Conv: |                    | 0.009%
/features/features.15/conv/conv.0/conv.0.0/Conv: |                    | 0.008%
/features/features.5/conv/conv.0/conv.0.0/Conv:  |                    | 0.006%
/features/features.6/conv/conv.0/conv.0.0/Conv:  |                    | 0.005%
/features/features.9/conv/conv.0/conv.0.0/Conv:  |                    | 0.003%
/features/features.18/features.18.0/Conv:        |                    | 0.002%
/features/features.10/conv/conv.0/conv.0.0/Conv: |                    | 0.002%
/features/features.8/conv/conv.0/conv.0.0/Conv:  |                    | 0.002%

* Prec@1 60.500 Prec@5 83.275*
```

Quantization error analysis

The top1 accuracy after quantization is only 60.5%, which is far from the accuracy of the float model (71.878%). The quantization model has a large loss in accuracy, including:

- **Graphwise Error**

  The last layer of the model is /classifier/classifier.1/Gemm, and the cumulative error of this layer is 25.591%. In experience, the cumulative error of the last layer is less than 10%, and the accuracy loss of the quantization model is small.

- **Layerwise error**

Observing the Layerwise error, it is found that the errors of most layers are less than 1%, indicating that the quantization errors of most layers are small, and only a few layers have large errors. We can choose to quantize the layers with large errors using int16. For details, please see mixed precision quantization.

Mixed precision quantization

esp-dl supports mixed precision quantization, which can be used to quantize some layers using int16 and some layers using int8. The quantization error of the model can be reduced by using mixed precision quantization.

Quantization settings

```
from esp_ppq.api import get_target_platform
target="esp32p4"
num_of_bits=8
batch_size=32

# The following layers are quantized using int16
quant_setting = QuantizationSettingFactory.espdl_setting()
quant_setting.dispatching_table.append("/features/features.1/conv/conv.0/conv.0.0/Conv
↪", get_target_platform(TARGET, 16))
quant_setting.dispatching_table.append("/features/features.1/conv/conv.0/conv.0.2/Clip
↪", get_target_platform(TARGET, 16))
```

Quantization results

```
Layer                                           | NOISE:SIGNAL POWER RATIO
/features/features.16/conv/conv.2/Conv:         | ████████████████████ | 31.585%
/features/features.15/conv/conv.2/Conv:         | ███████████████████  | 29.253%
/features/features.17/conv/conv.0/conv.0.0/Conv: | ████████████████     | 25.077%
/features/features.14/conv/conv.2/Conv:         | ████████████████     | 24.819%
/features/features.17/conv/conv.2/Conv:         | ██████████           | 19.546%
/features/features.13/conv/conv.2/Conv:         | ██████████           | 19.283%
/features/features.16/conv/conv.0/conv.0.0/Conv: | ██████████           | 18.764%
/features/features.16/conv/conv.1/conv.1.0/Conv: | ██████████           | 18.596%
/features/features.18/features.18.0/Conv:       | ██████████           | 18.541%
/features/features.15/conv/conv.0/conv.0.0/Conv: | ████████             | 15.633%
/features/features.12/conv/conv.2/Conv:         | ███████              | 14.784%
/features/features.15/conv/conv.1/conv.1.0/Conv: | ███████              | 14.773%
/features/features.14/conv/conv.1/conv.1.0/Conv: | ███████              | 13.700%
/features/features.6/conv/conv.2/Conv:          | ██████               | 12.824%
/features/features.10/conv/conv.2/Conv:         | ██████               | 11.727%
/features/features.14/conv/conv.0/conv.0.0/Conv: | ██████               | 10.612%
/features/features.11/conv/conv.2/Conv:         | ████                 | 10.262%
/features/features.9/conv/conv.2/Conv:          | ████                 | 9.967%
/classifier/classifier.1/Gemm:                  | ████                 | 9.117%
/features/features.5/conv/conv.2/Conv:          | ████                 | 8.915%
/features/features.7/conv/conv.2/Conv:          | ███                  | 8.690%
/features/features.3/conv/conv.2/Conv:          | ███                  | 8.586%
/features/features.4/conv/conv.2/Conv:          | ███                  | 7.525%
/features/features.13/conv/conv.1/conv.1.0/Conv: | ███                  | 7.432%
/features/features.12/conv/conv.1/conv.1.0/Conv: | ███                  | 7.317%
/features/features.13/conv/conv.0/conv.0.0/Conv: | ██                   | 6.848%
/features/features.8/conv/conv.2/Conv:          | ██                   | 6.711%
/features/features.10/conv/conv.1/conv.1.0/Conv: | ██                   | 6.100%
/features/features.8/conv/conv.1/conv.1.0/Conv:  | ██                   | 6.043%
/features/features.11/conv/conv.1/conv.1.0/Conv: | ██                   | 5.962%
/features/features.9/conv/conv.1/conv.1.0/Conv:  | ██                   | 5.873%
```

(continues on next page)

(continued from previous page)

```
/features/features.12/conv/conv.0/conv.0.0/Conv: | ████                | 5.833%
/features/features.7/conv/conv.0/conv.0.0/Conv:  | ████                | 5.832%
/features/features.11/conv/conv.0/conv.0.0/Conv: | ████                | 5.736%
/features/features.6/conv/conv.1/conv.1.0/Conv:  | ████                | 5.639%
/features/features.5/conv/conv.1/conv.1.0/Conv:  | ███                 | 5.017%
/features/features.10/conv/conv.0/conv.0.0/Conv: | ███                 | 4.963%
/features/features.17/conv/conv.1/conv.1.0/Conv: | ███                 | 4.870%
/features/features.3/conv/conv.1/conv.1.0/Conv:  | ███                 | 4.655%
/features/features.2/conv/conv.2/Conv:           | ███                 | 4.650%
/features/features.4/conv/conv.0/conv.0.0/Conv:  | ███                 | 4.648%
/features/features.1/conv/conv.1/Conv:           | ███                 | 4.318%
/features/features.9/conv/conv.0/conv.0.0/Conv:  | ██                  | 3.849%
/features/features.6/conv/conv.0/conv.0.0/Conv:  | ██                  | 3.712%
/features/features.4/conv/conv.1/conv.1.0/Conv:  | ██                  | 3.394%
/features/features.8/conv/conv.0/conv.0.0/Conv:  | ██                  | 3.391%
/features/features.7/conv/conv.1/conv.1.0/Conv:  | ██                  | 2.713%
/features/features.2/conv/conv.1/conv.1.0/Conv:  | ██                  | 2.637%
/features/features.2/conv/conv.0/conv.0.0/Conv:  | ██                  | 2.602%
/features/features.5/conv/conv.0/conv.0.0/Conv:  | ▌                   | 2.397%
/features/features.3/conv/conv.0/conv.0.0/Conv:  | ▌                   | 1.759%
/features/features.1/conv/conv.0/conv.0.0/Conv:  |                     | 0.433%
/features/features.0/features.0.0/Conv:          |                     | 0.119%
Analysing Layerwise quantization error:: 100
↪%|████████████████████████████████████████████████████████████████████████████████████████████████████████████
↪53/53 [08:27<00:00,  9.58s/it]
*
Layer                                            | NOISE:SIGNAL POWER RATIO
/features/features.1/conv/conv.1/Conv:           | ████████████████████ | 1.096%
/features/features.0/features.0.0/Conv:          | ███████████████      | 0.844%
/features/features.2/conv/conv.1/conv.1.0/Conv:  | ██████████           | 0.574%
/features/features.3/conv/conv.1/conv.1.0/Conv:  | ████████             | 0.425%
/features/features.15/conv/conv.1/conv.1.0/Conv: | ████▌                | 0.272%
/features/features.9/conv/conv.1/conv.1.0/Conv:  | ████                 | 0.238%
/features/features.17/conv/conv.1/conv.1.0/Conv: | ████                 | 0.214%
/features/features.4/conv/conv.1/conv.1.0/Conv:  | ███                  | 0.180%
/features/features.11/conv/conv.1/conv.1.0/Conv: | ███                  | 0.151%
/features/features.12/conv/conv.1/conv.1.0/Conv: | ███                  | 0.148%
/features/features.16/conv/conv.1/conv.1.0/Conv: | ███                  | 0.146%
/features/features.14/conv/conv.2/Conv:          | ██                   | 0.136%
/features/features.13/conv/conv.1/conv.1.0/Conv: | ██                   | 0.105%
/features/features.6/conv/conv.1/conv.1.0/Conv:  | ██                   | 0.105%
/features/features.8/conv/conv.1/conv.1.0/Conv:  | ▌                    | 0.083%
/features/features.5/conv/conv.1/conv.1.0/Conv:  | ▌                    | 0.076%
/features/features.3/conv/conv.2/Conv:           | ▌                    | 0.075%
/features/features.16/conv/conv.2/Conv:          | ▌                    | 0.074%
/features/features.13/conv/conv.0/conv.0.0/Conv: | ▌                    | 0.072%
/features/features.7/conv/conv.2/Conv:           | ▌                    | 0.071%
/features/features.15/conv/conv.2/Conv:          | ▌                    | 0.066%
/features/features.4/conv/conv.2/Conv:           | ▌                    | 0.065%
/features/features.11/conv/conv.2/Conv:          | ▌                    | 0.063%
/classifier/classifier.1/Gemm:                   | ▌                    | 0.063%
/features/features.13/conv/conv.2/Conv:          | ▌                    | 0.059%
/features/features.2/conv/conv.0/conv.0.0/Conv:  | ▌                    | 0.054%
/features/features.10/conv/conv.1/conv.1.0/Conv: | ▌                    | 0.042%
/features/features.17/conv/conv.0/conv.0.0/Conv: | ▌                    | 0.040%
/features/features.2/conv/conv.2/Conv:           | ▌                    | 0.038%
/features/features.4/conv/conv.0/conv.0.0/Conv:  | ▌                    | 0.034%
```

(continues on next page)

(continued from previous page)

```
/features/features.17/conv/conv.2/Conv:           | ▉                      | 0.030%
/features/features.14/conv/conv.0/conv.0.0/Conv:  |                        | 0.025%
/features/features.16/conv/conv.0/conv.0.0/Conv:  |                        | 0.024%
/features/features.10/conv/conv.2/Conv:           |                        | 0.022%
/features/features.11/conv/conv.0/conv.0.0/Conv:  |                        | 0.021%
/features/features.9/conv/conv.2/Conv:            |                        | 0.021%
/features/features.14/conv/conv.1/conv.1.0/Conv:  |                        | 0.020%
/features/features.7/conv/conv.1/conv.1.0/Conv:   |                        | 0.020%
/features/features.5/conv/conv.2/Conv:            |                        | 0.019%
/features/features.8/conv/conv.2/Conv:            |                        | 0.018%
/features/features.12/conv/conv.2/Conv:           |                        | 0.017%
/features/features.1/conv/conv.0/conv.0.0/Conv:   |                        | 0.017%
/features/features.6/conv/conv.2/Conv:            |                        | 0.014%
/features/features.7/conv/conv.0/conv.0.0/Conv:   |                        | 0.014%
/features/features.3/conv/conv.0/conv.0.0/Conv:   |                        | 0.013%
/features/features.12/conv/conv.0/conv.0.0/Conv:  |                        | 0.009%
/features/features.15/conv/conv.0/conv.0.0/Conv:  |                        | 0.008%
/features/features.5/conv/conv.0/conv.0.0/Conv:   |                        | 0.006%
/features/features.6/conv/conv.0/conv.0.0/Conv:   |                        | 0.005%
/features/features.9/conv/conv.0/conv.0.0/Conv:   |                        | 0.003%
/features/features.18/features.18.0/Conv:         |                        | 0.002%
/features/features.10/conv/conv.0/conv.0.0/Conv:  |                        | 0.002%
/features/features.8/conv/conv.0/conv.0.0/Conv:   |                        | 0.002%

* Prec@1 69.550 Prec@5 88.450*
```

Quantization Error Analysis

After replacing the layer with the largest error with 16-bit quantization, it can be observed that the model accuracy is significantly improved. The top1 accuracy after quantization is 69.550%, which is close to the accuracy of the float model (71.878%). The cumulative error of the last layer of the model `/classifier/classifier.1/Gemm` is 9.117%.

Layerwise equalization quantization

This method is proposed in the paper Data-Free Quantization Through Weight Equalization and Bias Correction. When using this method, the original ReLU6 in the MobilenetV2 model needs to be replaced with ReLU.

Quantization Settings

```python
import torch.nn as nn
def convert_relu6_to_relu(model):
    for child_name, child in model.named_children():
        if isinstance(child, nn.ReLU6):
            setattr(model, child_name, nn.ReLU())
        else:
            convert_relu6_to_relu(child)
    return model

# Replace ReLU6 with ReLU
model = convert_relu6_to_relu(model)
# Use layerwise equalization
quant_setting = QuantizationSettingFactory.espdl_setting()
quant_setting.equalization = True
quant_setting.equalization_setting.iterations = 4
quant_setting.equalization_setting.value_threshold = .4
```

(continues on next page)

(continued from previous page)

```
quant_setting.equalization_setting.opt_level = 2
quant_setting.equalization_setting.interested_layers = None
```

```
Layer                                                | NOISE:SIGNAL POWER RATIO
/features/features.16/conv/conv.2/Conv:              | ████████████████████ | 34.497%
/features/features.15/conv/conv.2/Conv:              | ██████████████████   | 30.813%
/features/features.14/conv/conv.2/Conv:              | ███████████████      | 25.876%
/features/features.17/conv/conv.0/conv.0.0/Conv:     | ██████████████       | 24.498%
/features/features.17/conv/conv.2/Conv:              | ████████████         | 20.290%
/features/features.13/conv/conv.2/Conv:              | ████████████         | 20.177%
/features/features.16/conv/conv.0/conv.0.0/Conv:     | ████████████         | 19.993%
/features/features.18/features.18.0/Conv:            | ███████████          | 19.536%
/features/features.16/conv/conv.1/conv.1.0/Conv:     | ██████████           | 17.879%
/features/features.12/conv/conv.2/Conv:              | ██████████           | 17.150%
/features/features.15/conv/conv.0/conv.0.0/Conv:     | █████████            | 15.970%
/features/features.15/conv/conv.1/conv.1.0/Conv:     | █████████            | 15.254%
/features/features.1/conv/conv.1/Conv:               | █████████            | 15.122%
/features/features.10/conv/conv.2/Conv:              | █████████            | 14.917%
/features/features.6/conv/conv.2/Conv:               | ████████             | 13.446%
/features/features.11/conv/conv.2/Conv:              | ███████              | 12.533%
/features/features.9/conv/conv.2/Conv:               | ███████              | 11.479%
/features/features.14/conv/conv.1/conv.1.0/Conv:     | ███████              | 11.470%
/features/features.5/conv/conv.2/Conv:               | ██████               | 10.669%
/features/features.3/conv/conv.2/Conv:               | ██████               | 10.526%
/features/features.14/conv/conv.0/conv.0.0/Conv:     | ██████               | 9.529%
/features/features.7/conv/conv.2/Conv:               | █████                | 9.500%
/classifier/classifier.1/Gemm:                       | █████                | 8.965%
/features/features.4/conv/conv.2/Conv:               | █████                | 8.674%
/features/features.12/conv/conv.1/conv.1.0/Conv:     | █████                | 8.349%
/features/features.13/conv/conv.1/conv.1.0/Conv:     | █████                | 8.068%
/features/features.8/conv/conv.2/Conv:               | █████                | 7.961%
/features/features.13/conv/conv.0/conv.0.0/Conv:     | ████                 | 7.451%
/features/features.10/conv/conv.1/conv.1.0/Conv:     | ████                 | 6.714%
/features/features.9/conv/conv.1/conv.1.0/Conv:      | ████                 | 6.399%
/features/features.8/conv/conv.1/conv.1.0/Conv:      | ████                 | 6.369%
/features/features.11/conv/conv.1/conv.1.0/Conv:     | ████                 | 6.222%
/features/features.2/conv/conv.2/Conv:               | ███                  | 5.867%
/features/features.5/conv/conv.1/conv.1.0/Conv:      | ███                  | 5.719%
/features/features.12/conv/conv.0/conv.0.0/Conv:     | ███                  | 5.546%
/features/features.6/conv/conv.1/conv.1.0/Conv:      | ███                  | 5.414%
/features/features.10/conv/conv.0/conv.0.0/Conv:     | ███                  | 5.093%
/features/features.17/conv/conv.1/conv.1.0/Conv:     | ███                  | 4.951%
/features/features.11/conv/conv.0/conv.0.0/Conv:     | ███                  | 4.941%
/features/features.2/conv/conv.1/conv.1.0/Conv:      | ███                  | 4.825%
/features/features.7/conv/conv.0/conv.0.0/Conv:      | ██                   | 4.330%
/features/features.2/conv/conv.0/conv.0.0/Conv:      | ██                   | 4.299%
/features/features.3/conv/conv.1/conv.1.0/Conv:      | ██                   | 4.283%
/features/features.4/conv/conv.0/conv.0.0/Conv:      | ██                   | 3.477%
/features/features.4/conv/conv.1/conv.1.0/Conv:      | ██                   | 3.287%
/features/features.8/conv/conv.0/conv.0.0/Conv:      | ██                   | 2.787%
/features/features.9/conv/conv.0/conv.0.0/Conv:      | ██                   | 2.774%
/features/features.6/conv/conv.0/conv.0.0/Conv:      | ██                   | 2.705%
/features/features.7/conv/conv.1/conv.1.0/Conv:      | ██                   | 2.636%
/features/features.5/conv/conv.0/conv.0.0/Conv:      | ▌                    | 1.846%
/features/features.3/conv/conv.0/conv.0.0/Conv:      | ▌                    | 1.170%
/features/features.1/conv/conv.0/conv.0.0/Conv:      |                      | 0.389%
```

(continues on next page)

(continued from previous page)

```
/features/features.0/features.0.0/Conv:          |                        | 0.025%
Analysing Layerwise quantization error:: 100%|██████████| 53/53 [07:46<00:00,  8.80s/
↪it]
Layer                                            | NOISE:SIGNAL POWER RATIO
/features/features.1/conv/conv.0/conv.0.0/Conv:  | ████████████████████   | 0.989%
/features/features.0/features.0.0/Conv:          | ████████████████       | 0.845%
/features/features.16/conv/conv.2/Conv:          | ████                   | 0.238%
/features/features.17/conv/conv.2/Conv:          | ███                    | 0.202%
/features/features.14/conv/conv.2/Conv:          | ███                    | 0.198%
/features/features.1/conv/conv.1/Conv:           | ███                    | 0.192%
/features/features.15/conv/conv.2/Conv:          | ██                     | 0.145%
/features/features.4/conv/conv.2/Conv:           | ▌                      | 0.120%
/features/features.2/conv/conv.2/Conv:           | ▌                      | 0.111%
/features/features.2/conv/conv.1/conv.1.0/Conv:  | ▌                      | 0.079%
/classifier/classifier.1/Gemm:                   | ▏                      | 0.062%
/features/features.13/conv/conv.2/Conv:          | ▏                      | 0.050%
/features/features.3/conv/conv.2/Conv:           | ▏                      | 0.050%
/features/features.12/conv/conv.2/Conv:          | ▏                      | 0.050%
/features/features.5/conv/conv.1/conv.1.0/Conv:  | ▏                      | 0.047%
/features/features.3/conv/conv.1/conv.1.0/Conv:  | ▏                      | 0.046%
/features/features.7/conv/conv.2/Conv:           | ▏                      | 0.045%
/features/features.5/conv/conv.2/Conv:           | ▏                      | 0.030%
/features/features.11/conv/conv.2/Conv:          | ▏                      | 0.028%
/features/features.6/conv/conv.2/Conv:           | ▏                      | 0.027%
/features/features.6/conv/conv.1/conv.1.0/Conv:  | ▏                      | 0.026%
/features/features.4/conv/conv.0/conv.0.0/Conv:  |                        | 0.025%
/features/features.15/conv/conv.1/conv.1.0/Conv: |                        | 0.023%
/features/features.8/conv/conv.1/conv.1.0/Conv:  |                        | 0.021%
/features/features.10/conv/conv.2/Conv:          |                        | 0.020%
/features/features.11/conv/conv.1/conv.1.0/Conv: |                        | 0.020%
/features/features.16/conv/conv.1/conv.1.0/Conv: |                        | 0.017%
/features/features.14/conv/conv.0/conv.0.0/Conv: |                        | 0.016%
/features/features.4/conv/conv.1/conv.1.0/Conv:  |                        | 0.012%
/features/features.13/conv/conv.1/conv.1.0/Conv: |                        | 0.012%
/features/features.13/conv/conv.0/conv.0.0/Conv: |                        | 0.012%
/features/features.12/conv/conv.1/conv.1.0/Conv: |                        | 0.012%
/features/features.17/conv/conv.0/conv.0.0/Conv: |                        | 0.011%
/features/features.12/conv/conv.0/conv.0.0/Conv: |                        | 0.011%
/features/features.2/conv/conv.0/conv.0.0/Conv:  |                        | 0.010%
/features/features.9/conv/conv.2/Conv:           |                        | 0.008%
/features/features.8/conv/conv.2/Conv:           |                        | 0.008%
/features/features.10/conv/conv.1/conv.1.0/Conv: |                        | 0.008%
/features/features.16/conv/conv.0/conv.0.0/Conv: |                        | 0.008%
/features/features.7/conv/conv.0/conv.0.0/Conv:  |                        | 0.008%
/features/features.10/conv/conv.0/conv.0.0/Conv: |                        | 0.006%
/features/features.15/conv/conv.0/conv.0.0/Conv: |                        | 0.005%
/features/features.3/conv/conv.0/conv.0.0/Conv:  |                        | 0.004%
/features/features.11/conv/conv.0/conv.0.0/Conv: |                        | 0.004%
/features/features.18/features.18.0/Conv:        |                        | 0.003%
/features/features.5/conv/conv.0/conv.0.0/Conv:  |                        | 0.003%
/features/features.9/conv/conv.1/conv.1.0/Conv:  |                        | 0.003%
/features/features.6/conv/conv.0/conv.0.0/Conv:  |                        | 0.003%
/features/features.7/conv/conv.1/conv.1.0/Conv:  |                        | 0.003%
/features/features.17/conv/conv.1/conv.1.0/Conv: |                        | 0.002%
/features/features.14/conv/conv.1/conv.1.0/Conv: |                        | 0.002%
/features/features.8/conv/conv.0/conv.0.0/Conv:  |                        | 0.001%
/features/features.9/conv/conv.0/conv.0.0/Conv:  |                        | 0.001%
```

(continues on next page)

(continued from previous page)

```
* Prec@1 69.800 Prec@5 88.550
```

Quantization Error Analysis

Note that applying layerwise equalization to 8-bit quantization helps reduce quantization loss. The cumulative error of the last layer of the model `/classifier/classifier.1/Gemm` is 8.965%. The top1 accuracy after quantization is 69.800%, which is closer to the accuracy of the float model (71.878%) and higher than the quantization accuracy of mixed precision quantization.

Note: To further reduce the quantization error, you can try using QAT (Auantization Aware Training). For specific methods, please refer to PPQ QAT example.

3.6.3 Model deployment

examples

Image classification base class

- dl_cls_base.hpp
- dl_cls_base.cpp

Pre-process

`ImagePreprocessor` class contains the common pre-precoess pipeline, `color conversion`, `crop`, `resize`, `normalization`, `quantize`

- dl_image_preprocessor.hpp
- dl_image_preprocessor.cpp

Post-process

- dl_cls_postprocessor.hpp
- dl_cls_postprocessor.cpp
- imagenet_cls_postprocessor.hpp
- imagenet_cls_postprocessor.cpp

3.7 How to deploy YOLO11n

In this tutorial, we will introduce how to quantize a pre-trained YOLO11n model using ESP-PPQ and deploy the quantized YOLO11n model using ESP-DL.

3.7.1 Preparation

1. *ESP_IDF*
2. *ESP_PPQ*

3.7.2 Model quantization

Pre-trained Model

You can download pre-trained yolo11n model from Ultralytics release.

Currently, ESP-PPQ supports ONNX, PyTorch, and TensorFlow models. During the quantization process, PyTorch and TensorFlow models are first converted to ONNX models, so the pre-trained yolo11n model needs to be converted to an ONNX model.

Specifically, refer to the script export_onnx.py to convert the pre-trained yolo11n model to an ONNX model.

In the script, we have overridden the forward method of the Detect class, which offers following advantages:

- Faster inference. Compared to the original yolo11n model, operations related to decoding bounding boxes in Detect head are moved from the inference pass to the post-processing phase, resulting in a significant reduction in inference latency. On one hand, operations like `Conv`, `Transpose`, `Slice`, `Split` and `Concat` are time-consuming when applied during inference pass. On the other hand, the inference outputs are first filtered using a score threshold before decoding the boxes in the post-processing pass, which significantly reduces the number of calculations, thereby accelerating the overall inference speed.
- Lower quantization Error. The `Concat` and `Add` operators adopt joint quantization in ESP-PPQ. To reduce quantization errors, the box and score are output by separate branches, rather than being concatenated, due to

the significant difference in their ranges. Similarly, since the ranges of the two inputs of `Add` and `Sub` differ significantly, the calculations are performed in the post-processing phase to avoid quantization errors.

Calibration Dataset

The calibration dataset needs to match the input format of the model. The calibration dataset should cover all possible input scenarios to better quantize the model. Here, the calibration dataset used in this example is calib_yolo11n.

8bit default configuration quantization

Quantization settings

```
target="esp32p4"
num_of_bits=8
batch_size=32
quant_setting = QuantizationSettingFactory.espdl_setting() # default setting
```

Quantization results

```
Layer                                          | NOISE:SIGNAL POWER RATIO
/model.10/m/m.0/ffn/ffn.1/conv/Conv:           | ████████████████████ | 36.008%
/model.10/m/m.0/attn/proj/conv/Conv:           | ██████████████       | 28.705%
/model.23/cv3.2/cv3.2.0/cv3.2.0.0/conv/Conv:   | ████████████         | 22.865%
/model.23/cv2.2/cv2.2.0/conv/Conv:             | ███████████          | 21.718%
/model.23/cv3.2/cv3.2.1/cv3.2.1.1/conv/Conv:   | ███████████          | 21.624%
/model.23/cv2.2/cv2.2.1/conv/Conv:             | ███████████          | 21.392%
/model.23/cv3.2/cv3.2.0/cv3.2.0.1/conv/Conv:   | ███████████          | 21.224%
/model.22/m.0/cv2/conv/Conv:                   | ██████████           | 19.763%
/model.23/cv3.0/cv3.0.1/cv3.0.1.1/conv/Conv:   | ██████████           | 19.436%
/model.22/m.0/cv3/conv/Conv:                   | ██████████           | 19.378%
/model.23/cv3.1/cv3.1.1/cv3.1.1.1/conv/Conv:   | █████████            | 18.913%
/model.22/m.0/m/m.1/cv2/conv/Conv:             | █████████            | 18.645%
/model.22/cv2/conv/Conv:                       | █████████            | 18.628%
/model.23/cv2.1/cv2.1.1/conv/Conv:             | █████████            | 17.980%
/model.8/m.0/cv2/conv/Conv:                    | ████████             | 16.247%
/model.23/cv2.0/cv2.0.1/conv/Conv:             | ████████             | 15.602%
/model.10/m/m.0/attn/qkv/conv/Conv:            | ███████              | 14.666%
/model.10/m/m.0/attn/pe/conv/Conv:             | ███████              | 14.556%
/model.23/cv2.1/cv2.1.0/conv/Conv:             | ███████              | 14.302%
/model.22/cv1/conv/Conv:                       | ███████              | 13.921%
/model.10/m/m.0/attn/MatMul_1:                 | ███████              | 13.905%
/model.10/cv1/conv/Conv:                       | ██████               | 13.494%
/model.23/cv3.1/cv3.1.0/cv3.1.0.1/conv/Conv:   | ██████               | 11.800%
/model.19/m.0/cv2/conv/Conv:                   | ██████               | 11.515%
/model.22/m.0/m/m.0/cv2/conv/Conv:             | ██████               | 11.286%
/model.20/conv/Conv:                           | ██████               | 10.930%
/model.13/m.0/cv2/conv/Conv:                   | ██████               | 10.882%
/model.23/cv3.2/cv3.2.1/cv3.2.1.0/conv/Conv:   | ██████               | 10.692%
/model.23/cv2.2/cv2.2.2/Conv:                  | ██████               | 10.113%
/model.10/cv2/conv/Conv:                       | ████                 | 9.720%
/model.8/cv2/conv/Conv:                        | ████                 | 9.598%
/model.8/m.0/cv1/conv/Conv:                    | ████                 | 9.470%
/model.19/cv2/conv/Conv:                       | ████                 | 9.314%
/model.22/m.0/m/m.0/cv1/conv/Conv:             | ████                 | 9.068%
/model.23/cv3.0/cv3.0.0/cv3.0.0.1/conv/Conv:   | ████                 | 9.065%
```

(continues on next page)

(continued from previous page)

```
/model.8/cv1/conv/Conv:                          | ████▌                | 9.051%
/model.8/m.0/cv3/conv/Conv:                      | ████▍                | 9.044%
/model.6/m.0/cv2/conv/Conv:                      | ████▎                | 8.811%
/model.22/m.0/m/m.1/cv1/conv/Conv:               | ████▎                | 8.781%
/model.13/cv2/conv/Conv:                         | ████▏                | 8.687%
/model.8/m.0/m/m.0/cv1/conv/Conv:                | ████▏                | 8.503%
/model.8/m.0/m/m.0/cv2/conv/Conv:                | ████                 | 8.470%
/model.19/cv1/conv/Conv:                         | ████                 | 8.199%
/model.10/m/m.0/attn/MatMul:                     | ████                 | 8.117%
/model.8/m.0/m/m.1/cv1/conv/Conv:                | ███▉                 | 7.964%
/model.13/cv1/conv/Conv:                         | ███▋                 | 7.734%
/model.19/m.0/cv1/conv/Conv:                     | ███▋                 | 7.661%
/model.22/m.0/cv1/conv/Conv:                     | ███▌                 | 7.490%
/model.13/m.0/cv1/conv/Conv:                     | ███▍                 | 7.162%
/model.8/m.0/m/m.1/cv2/conv/Conv:                | ███▍                 | 7.145%
/model.23/cv2.0/cv2.0.0/conv/Conv:               | ███▎                 | 7.041%
/model.23/cv2.1/cv2.1.2/Conv:                    | ███▎                 | 6.917%
/model.23/cv2.0/cv2.0.2/Conv:                    | ███▏                 | 6.778%
/model.23/cv3.1/cv3.1.1/cv3.1.1.0/conv/Conv:     | ███                  | 6.641%
/model.17/conv/Conv:                             | ██▉                  | 6.125%
/model.16/m.0/cv2/conv/Conv:                     | ██▊                  | 5.937%
/model.6/cv2/conv/Conv:                          | ██▊                  | 5.838%
/model.6/m.0/cv3/conv/Conv:                      | ██▊                  | 5.832%
/model.6/cv1/conv/Conv:                          | ██▋                  | 5.688%
/model.7/conv/Conv:                              | ██▋                  | 5.612%
/model.9/cv2/conv/Conv:                          | ██▌                  | 5.367%
/model.10/m/m.0/ffn/ffn.0/conv/Conv:             | ██▍                  | 5.158%
/model.6/m.0/m/m.0/cv1/conv/Conv:                | ██▍                  | 5.143%
/model.16/m.0/cv1/conv/Conv:                     | ██▍                  | 5.137%
/model.23/cv3.1/cv3.1.0/cv3.1.0.0/conv/Conv:     | ██▍                  | 5.087%
/model.16/cv2/conv/Conv:                         | ██▎                  | 4.989%
/model.2/cv2/conv/Conv:                          | ██                   | 4.547%
/model.6/m.0/m/m.0/cv2/conv/Conv:                | ██                   | 4.441%
/model.23/cv3.0/cv3.0.1/cv3.0.1.0/conv/Conv:     | ██                   | 4.343%
/model.3/conv/Conv:                              | ██                   | 4.304%
/model.6/m.0/m/m.1/cv1/conv/Conv:                | ██                   | 4.006%
/model.5/conv/Conv:                              | ██                   | 3.932%
/model.6/m.0/cv1/conv/Conv:                      | ██                   | 3.837%
/model.4/cv1/conv/Conv:                          | ██                   | 3.687%
/model.2/cv1/conv/Conv:                          | ██                   | 3.565%
/model.4/cv2/conv/Conv:                          | ██                   | 3.559%
/model.16/cv1/conv/Conv:                         | ██                   | 3.107%
/model.2/m.0/cv2/conv/Conv:                      | ██                   | 2.882%
/model.6/m.0/m/m.1/cv2/conv/Conv:                | ▉                    | 2.758%
/model.4/m.0/cv1/conv/Conv:                      | ▉                    | 2.564%
/model.9/cv1/conv/Conv:                          | ▉                    | 2.017%
/model.4/m.0/cv2/conv/Conv:                      | ▉                    | 1.785%
/model.23/cv3.0/cv3.0.0/cv3.0.0.0/conv/Conv:     | ▉                    | 1.327%
/model.1/conv/Conv:                              | ▉                    | 1.313%
/model.23/cv3.2/cv3.2.2/Conv:                    | ▉                    | 1.155%
/model.2/m.0/cv1/conv/Conv:                      |                      | 0.727%
/model.23/cv3.1/cv3.1.2/Conv:                    |                      | 0.493%
/model.23/cv3.0/cv3.0.2/Conv:                    |                      | 0.282%
/model.0/conv/Conv:                              |                      | 0.159%
Analysing Layerwise quantization error:: 100%|██████████| 89/89 [03:39<00:00,  2.46s/
↪it]
Layer                                            | NOISE:SIGNAL POWER RATIO
```

(continues on next page)

(continued from previous page)

```
/model.1/conv/Conv:                          | ████████████████████ | 0.384%
/model.22/cv1/conv/Conv:                     | ████████████         | 0.247%
/model.4/cv2/conv/Conv:                      | ████████████         | 0.233%
/model.2/cv2/conv/Conv:                      | ██████████           | 0.201%
/model.0/conv/Conv:                          | ██████████           | 0.192%
/model.9/cv2/conv/Conv:                      | ████████             | 0.156%
/model.10/cv1/conv/Conv:                     | ███████              | 0.132%
/model.3/conv/Conv:                          | ██████               | 0.108%
/model.4/cv1/conv/Conv:                      | ████                 | 0.074%
/model.16/cv1/conv/Conv:                     | ███                  | 0.066%
/model.2/cv1/conv/Conv:                      | ███                  | 0.060%
/model.23/cv2.0/cv2.0.0/conv/Conv:           | ███                  | 0.052%
/model.2/m.0/cv1/conv/Conv:                  | ██                   | 0.044%
/model.6/cv1/conv/Conv:                      | ██                   | 0.033%
/model.10/m/m.0/attn/pe/conv/Conv:           | ██                   | 0.029%
/model.2/m.0/cv2/conv/Conv:                  | █                    | 0.028%
/model.22/m.0/m/m.0/cv1/conv/Conv:           | █                    | 0.023%
/model.16/cv2/conv/Conv:                     | █                    | 0.021%
/model.16/m.0/cv2/conv/Conv:                 | █                    | 0.020%
/model.19/m.0/cv1/conv/Conv:                 | █                    | 0.020%
/model.4/m.0/cv1/conv/Conv:                  | █                    | 0.018%
/model.19/cv2/conv/Conv:                     | █                    | 0.017%
/model.4/m.0/cv2/conv/Conv:                  | █                    | 0.016%
/model.10/m/m.0/attn/qkv/conv/Conv:          | █                    | 0.016%
/model.19/cv1/conv/Conv:                     | █                    | 0.015%
/model.13/cv2/conv/Conv:                     | █                    | 0.015%
/model.8/cv1/conv/Conv:                      | █                    | 0.013%
/model.23/cv2.1/cv2.1.0/conv/Conv:           | █                    | 0.013%
/model.23/cv2.2/cv2.2.1/conv/Conv:           | █                    | 0.012%
/model.13/cv1/conv/Conv:                     | █                    | 0.012%
/model.10/cv2/conv/Conv:                     | █                    | 0.011%
/model.13/m.0/cv1/conv/Conv:                 | █                    | 0.011%
/model.6/cv2/conv/Conv:                      | █                    | 0.011%
/model.13/m.0/cv2/conv/Conv:                 | █                    | 0.010%
/model.5/conv/Conv:                          |                      | 0.010%
/model.19/m.0/cv2/conv/Conv:                 |                      | 0.009%
/model.6/m.0/m/m.1/cv1/conv/Conv:            |                      | 0.009%
/model.23/cv3.0/cv3.0.0/cv3.0.0.1/conv/Conv: |                      | 0.008%
/model.23/cv2.2/cv2.2.0/conv/Conv:           |                      | 0.008%
/model.23/cv2.1/cv2.1.1/conv/Conv:           |                      | 0.008%
/model.9/cv1/conv/Conv:                      |                      | 0.008%
/model.23/cv2.0/cv2.0.1/conv/Conv:           |                      | 0.007%
/model.16/m.0/cv1/conv/Conv:                 |                      | 0.007%
/model.17/conv/Conv:                         |                      | 0.007%
/model.23/cv3.1/cv3.1.1/cv3.1.1.0/conv/Conv: |                      | 0.007%
/model.10/m/m.0/ffn/ffn.1/conv/Conv:         |                      | 0.007%
/model.23/cv2.0/cv2.0.2/Conv:                |                      | 0.006%
/model.8/m.0/cv1/conv/Conv:                  |                      | 0.006%
/model.23/cv2.2/cv2.2.2/Conv:                |                      | 0.005%
/model.23/cv2.1/cv2.1.2/Conv:                |                      | 0.005%
/model.22/m.0/cv3/conv/Conv:                 |                      | 0.005%
/model.23/cv3.1/cv3.1.0/cv3.1.0.1/conv/Conv: |                      | 0.005%
/model.7/conv/Conv:                          |                      | 0.005%
/model.8/cv2/conv/Conv:                      |                      | 0.004%
/model.22/cv2/conv/Conv:                     |                      | 0.004%
/model.6/m.0/cv3/conv/Conv:                  |                      | 0.004%
/model.10/m/m.0/ffn/ffn.0/conv/Conv:         |                      | 0.004%
```

(continues on next page)

(continued from previous page)

```
/model.8/m.0/m/m.1/cv2/conv/Conv:             |                       | 0.004%
/model.22/m.0/m/m.1/cv1/conv/Conv:            |                       | 0.004%
/model.8/m.0/m/m.1/cv1/conv/Conv:             |                       | 0.004%
/model.23/cv3.1/cv3.1.1/cv3.1.1.1/conv/Conv: |                       | 0.003%
/model.10/m/m.0/attn/proj/conv/Conv:          |                       | 0.003%
/model.22/m.0/m/m.0/cv2/conv/Conv:            |                       | 0.003%
/model.22/m.0/cv1/conv/Conv:                  |                       | 0.003%
/model.8/m.0/cv3/conv/Conv:                   |                       | 0.003%
/model.6/m.0/m/m.0/cv1/conv/Conv:             |                       | 0.003%
/model.23/cv3.0/cv3.0.0/cv3.0.0.0/conv/Conv: |                       | 0.003%
/model.23/cv3.2/cv3.2.1/cv3.2.1.0/conv/Conv: |                       | 0.002%
/model.6/m.0/m/m.1/cv2/conv/Conv:             |                       | 0.002%
/model.8/m.0/m/m.0/cv2/conv/Conv:             |                       | 0.002%
/model.23/cv3.2/cv3.2.1/cv3.2.1.1/conv/Conv: |                       | 0.002%
/model.10/m/m.0/attn/MatMul_1:                |                       | 0.002%
/model.22/m.0/m/m.1/cv2/conv/Conv:            |                       | 0.001%
/model.6/m.0/m/m.0/cv2/conv/Conv:             |                       | 0.001%
/model.23/cv3.0/cv3.0.1/cv3.0.1.0/conv/Conv: |                       | 0.001%
/model.8/m.0/m/m.0/cv1/conv/Conv:             |                       | 0.001%
/model.23/cv3.2/cv3.2.0/cv3.2.0.1/conv/Conv: |                       | 0.001%
/model.23/cv3.0/cv3.0.1/cv3.0.1.1/conv/Conv: |                       | 0.001%
/model.6/m.0/cv1/conv/Conv:                   |                       | 0.001%
/model.23/cv3.2/cv3.2.2/Conv:                 |                       | 0.001%
/model.20/conv/Conv:                          |                       | 0.001%
/model.23/cv3.1/cv3.1.2/Conv:                 |                       | 0.001%
/model.23/cv3.2/cv3.2.0/cv3.2.0.0/conv/Conv: |                       | 0.001%
/model.6/m.0/cv2/conv/Conv:                   |                       | 0.001%
/model.23/cv3.0/cv3.0.2/Conv:                 |                       | 0.000%
/model.10/m/m.0/attn/MatMul:                  |                       | 0.000%
/model.23/cv3.1/cv3.1.0/cv3.1.0.0/conv/Conv: |                       | 0.000%
/model.8/m.0/cv2/conv/Conv:                   |                       | 0.000%
/model.22/m.0/cv2/conv/Conv:                  |                       | 0.000%
```

Quantization error analysis

With the same inputs, The mAP50:95 on COCO val2017 after quantization is only 30.7%, which is lower than that of the float model. There is a accuracy loss with:

- **Graphwise Error**

  The output layers of the model are /model.23/cv3.2/cv3.2.2/Conv, /model.23/cv2.2/cv2.2.2/Conv, /model.23/cv3.1/cv3.1.2/Conv, /model.23/cv2.1/cv2.1.2/Conv, /model.23/cv3.0/cv3.0.2/Conv and /model.23/cv2.0/cv2.0.2/Conv. The cumulative error for these layers are 1.155%, 10.113%, 0.493%, 6.917%, 0.282% and 6.778% respectively. Generally, if the cumulative error of the output layer is less than 10%, the loss in accuracy of the quantized model is minimal.

- **Layerwise error**

  Observing the Layerwise error, it is found that the errors for all layers are below 1%, indicating that the quantization errors for all layers are small.

We noticed that although the layer-wise errors for all layers are small, the cumulative errors in some layers are relatively large. This may be related to the complex CSP structure in the yolo11n model, where the inputs to the `Concat` or `Add` layers may have different distributions or scales. We can choose to quantize certain layers using int16 and optimize the quantization with horizontal layer split pass. For more details, please refer to the mixed-precision + horizontal layer split pass quantization test.

Mixed-Precision + Horizontal Layer Split Quantization

Splitting convolution layers or GEMM layers can reduce quantization error for better performance.

Quantization settings

```
from esp_ppq.api import get_target_platform
target="esp32p4"
num_of_bits=8
batch_size=32

# Quantize the following layers with 16-bits
quant_setting = QuantizationSettingFactory.espdl_setting()
quant_setting.dispatching_table.append("/model.2/cv2/conv/Conv", get_target_
↪platform(TARGET, 16))
quant_setting.dispatching_table.append("/model.3/conv/Conv", get_target_
↪platform(TARGET, 16))
quant_setting.dispatching_table.append("/model.4/cv2/conv/Conv", get_target_
↪platform(TARGET, 16))

# Horizontal Layer Split Pass
quant_setting.weight_split = True
quant_setting.weight_split_setting.method = 'balance'
quant_setting.weight_split_setting.value_threshold = 1.5
quant_setting.weight_split_setting.interested_layers = ['/model.0/conv/Conv', '/model.
↪1/conv/Conv']
```

Quantization results

```
Layer                                          | NOISE:SIGNAL POWER RATIO
/model.10/m/m.0/ffn/ffn.1/conv/Conv:           | ████████████████████ | 24.835%
/model.10/m/m.0/attn/proj/conv/Conv:           | ████████████████     | 18.632%
/model.23/cv2.2/cv2.2.1/conv/Conv:             | ███████████████      | 17.908%
/model.23/cv3.2/cv3.2.0/cv3.2.0.0/conv/Conv:   | ███████████████      | 16.922%
/model.23/cv2.2/cv2.2.0/conv/Conv:             | ██████████████       | 16.754%
/model.22/m.0/cv3/conv/Conv:                   | ████████████         | 15.404%
/model.23/cv3.2/cv3.2.0/cv3.2.0.1/conv/Conv:   | ████████████         | 15.042%
/model.23/cv3.0/cv3.0.1/cv3.0.1.1/conv/Conv:   | ████████████         | 14.948%
/model.22/m.0/m/m.1/cv2/conv/Conv:             | ████████████         | 14.702%
/model.23/cv3.2/cv3.2.1/cv3.2.1.1/conv/Conv:   | ███████████          | 13.683%
/model.22/cv2/conv/Conv:                       | ███████████          | 13.654%
/model.22/m.0/cv2/conv/Conv:                   | ███████████          | 13.514%
/model.23/cv3.1/cv3.1.1/cv3.1.1.1/conv/Conv:   | ██████████           | 12.885%
/model.23/cv2.1/cv2.1.1/conv/Conv:             | ████████             | 10.865%
/model.23/cv2.0/cv2.0.1/conv/Conv:             | ███████              | 9.875%
/model.23/cv2.1/cv2.1.0/conv/Conv:             | ███████              | 9.658%
/model.22/cv1/conv/Conv:                       | ██████               | 8.917%
/model.10/m/m.0/attn/MatMul_1:                 | ██████               | 8.368%
/model.23/cv2.2/cv2.2.2/Conv:                  | ██████               | 8.156%
/model.22/m.0/m/m.0/cv2/conv/Conv:             | ██████               | 8.056%
/model.10/m/m.0/attn/qkv/conv/Conv:            | ██████               | 7.948%
/model.23/cv3.1/cv3.1.0/cv3.1.0.1/conv/Conv:   | ██████               | 7.824%
/model.13/m.0/cv2/conv/Conv:                   | ██████               | 7.504%
/model.19/m.0/cv2/conv/Conv:                   | ██████               | 7.290%
/model.20/conv/Conv:                           | ██████               | 6.986%
/model.10/m/m.0/attn/pe/conv/Conv:             | ██████               | 6.926%
/model.23/cv3.0/cv3.0.0/cv3.0.0.1/conv/Conv:   | ████                 | 6.771%
/model.23/cv3.2/cv3.2.1/cv3.2.1.0/conv/Conv:   | ████                 | 6.756%
```

(continues on next page)

(continued from previous page)

```
/model.22/m.0/m/m.1/cv1/conv/Conv:           | ████▌                | 6.465%
/model.22/m.0/m/m.0/cv1/conv/Conv:           | ████▎                | 6.274%
/model.19/cv2/conv/Conv:                     | ████▎                | 6.116%
/model.10/cv1/conv/Conv:                     | ████▏                | 5.868%
/model.13/cv2/conv/Conv:                     | ████▏                | 5.815%
/model.10/cv2/conv/Conv:                     | ███▉                 | 5.664%
/model.19/cv1/conv/Conv:                     | ███▋                 | 5.178%
/model.8/m.0/cv2/conv/Conv:                  | ███▌                 | 4.970%
/model.19/m.0/cv1/conv/Conv:                 | ███▍                 | 4.919%
/model.23/cv3.1/cv3.1.1/cv3.1.1.0/conv/Conv: | ███▍                 | 4.864%
/model.22/m.0/cv1/conv/Conv:                 | ███▍                 | 4.844%
/model.10/m/m.0/attn/MatMul:                 | ███▎                 | 4.650%
/model.13/cv1/conv/Conv:                     | ███▎                 | 4.564%
/model.23/cv2.0/cv2.0.0/conv/Conv:           | ███                  | 4.389%
/model.13/m.0/cv1/conv/Conv:                 | ███                  | 4.243%
/model.23/cv2.0/cv2.0.2/Conv:                | ███                  | 4.232%
/model.23/cv2.1/cv2.1.2/Conv:                | ███                  | 4.222%
/model.6/m.0/cv2/conv/Conv:                  | ███                  | 4.023%
/model.17/conv/Conv:                         | ███                  | 3.754%
/model.16/m.0/cv2/conv/Conv:                 | ███                  | 3.511%
/model.8/m.0/cv1/conv/Conv:                  | ███                  | 3.277%
/model.16/m.0/cv1/conv/Conv:                 | ██▊                  | 3.158%
/model.23/cv3.0/cv3.0.1/cv3.0.1.0/conv/Conv: | ██▊                  | 3.155%
/model.23/cv3.1/cv3.1.0/cv3.1.0.0/conv/Conv: | ██▊                  | 3.152%
/model.8/cv2/conv/Conv:                      | ██▋                  | 3.119%
/model.8/m.0/m/m.1/cv1/conv/Conv:            | ██▋                  | 3.106%
/model.8/m.0/cv3/conv/Conv:                  | ██▋                  | 3.083%
/model.6/m.0/cv3/conv/Conv:                  | ██▋                  | 3.068%
/model.8/cv1/conv/Conv:                      | ██▋                  | 3.035%
/model.16/cv2/conv/Conv:                     | ██▋                  | 3.002%
/model.2/cv2/conv/Conv:                      | ██▋                  | 2.992%
/model.8/m.0/m/m.0/cv2/conv/Conv:            | ██▋                  | 2.971%
/model.6/cv1/conv/Conv:                      | ██▍                  | 2.819%
/model.8/m.0/m/m.0/cv1/conv/Conv:            | ██▍                  | 2.809%
/model.10/m/m.0/ffn/ffn.0/conv/Conv:         | ██▍                  | 2.760%
/model.2/cv1/conv/Conv:                      | ██▍                  | 2.683%
/model.6/cv2/conv/Conv:                      | ██▍                  | 2.630%
/model.8/m.0/m/m.1/cv2/conv/Conv:            | ██▍                  | 2.615%
/model.9/cv2/conv/Conv:                      | ██▍                  | 2.540%
/model.3/conv/Conv:                          | ██▍                  | 2.503%
/model.2/m.0/cv2/conv/Conv:                  | ██▍                  | 2.474%
/model.6/m.0/m/m.0/cv1/conv/Conv:            | ██▍                  | 2.273%
/model.6/m.0/m/m.0/cv2/conv/Conv:            | ██▍                  | 2.246%
/model.4/cv2/conv/Conv:                      | ██▍                  | 2.141%
/model.7/conv/Conv:                          | ██▍                  | 2.120%
/model.6/m.0/m/m.1/cv1/conv/Conv:            | ██▍                  | 2.069%
/model.5/conv/Conv:                          | ██▍                  | 2.015%
/model.16/cv1/conv/Conv:                     | ▉                    | 1.894%
/model.4/cv1/conv/Conv:                      | ▉                    | 1.793%
/model.4/m.0/cv1/conv/Conv:                  | ▉                    | 1.776%
/model.6/m.0/cv1/conv/Conv:                  | ▉                    | 1.731%
/model.6/m.0/m/m.1/cv2/conv/Conv:            | ▉                    | 1.550%
/model.4/m.0/cv2/conv/Conv:                  | ▉                    | 1.257%
/model.23/cv3.0/cv3.0.0/cv3.0.0.0/conv/Conv: | ▊                    | 0.886%
/model.1/conv/Conv:                          | ▋                    | 0.775%
/model.23/cv3.2/cv3.2.2/Conv:                | ▋                    | 0.771%
PPQ_Operation_2:                             |                      | 0.696%
```

(continues on next page)

(continued from previous page)

```
/model.9/cv1/conv/Conv:                       |                       | 0.695%
/model.2/m.0/cv1/conv/Conv:                   |                       | 0.534%
/model.23/cv3.1/cv3.1.2/Conv:                 |                       | 0.339%
/model.23/cv3.0/cv3.0.2/Conv:                 |                       | 0.190%
PPQ_Operation_0:                              |                       | 0.110%
/model.0/conv/Conv:                           |                       | 0.099%
Analysing Layerwise quantization error:: 100%|██████████| 91/91 [04:13<00:00,  2.79s/
↪it]
Layer                                         | NOISE:SIGNAL POWER RATIO
/model.22/cv1/conv/Conv:                      | ████████████████████  | 0.244%
/model.9/cv2/conv/Conv:                       | ████████████          | 0.156%
/model.10/cv1/conv/Conv:                      | ███████████           | 0.132%
/model.1/conv/Conv:                           | ██████                | 0.077%
/model.4/cv1/conv/Conv:                       | ██████                | 0.074%
/model.16/cv1/conv/Conv:                      | █████                 | 0.066%
/model.0/conv/Conv:                           | █████                 | 0.061%
/model.2/cv1/conv/Conv:                       | █████                 | 0.060%
/model.23/cv2.0/cv2.0.0/conv/Conv:            | ████                  | 0.052%
PPQ_Operation_0:                              | ████                  | 0.047%
/model.2/m.0/cv1/conv/Conv:                   | ████                  | 0.045%
/model.10/m/m.0/attn/pe/conv/Conv:            | ██                    | 0.029%
/model.2/m.0/cv2/conv/Conv:                   | ██                    | 0.029%
/model.10/m/m.0/attn/MatMul:                  | ██                    | 0.025%
/model.6/cv1/conv/Conv:                       | ██                    | 0.025%
/model.22/m.0/m/m.0/cv1/conv/Conv:            | ██                    | 0.023%
/model.16/cv2/conv/Conv:                      | ██                    | 0.021%
/model.16/m.0/cv2/conv/Conv:                  | ██                    | 0.020%
/model.19/m.0/cv1/conv/Conv:                  | ██                    | 0.020%
/model.4/m.0/cv1/conv/Conv:                   | ▉                     | 0.018%
/model.19/cv2/conv/Conv:                      | ▉                     | 0.017%
/model.4/m.0/cv2/conv/Conv:                   | ▉                     | 0.016%
/model.10/m/m.0/attn/qkv/conv/Conv:           | ▉                     | 0.016%
/model.19/cv1/conv/Conv:                      | ▉                     | 0.015%
/model.13/cv2/conv/Conv:                      | ▉                     | 0.015%
/model.23/cv2.1/cv2.1.0/conv/Conv:            | ▉                     | 0.013%
/model.23/cv2.2/cv2.2.1/conv/Conv:            | ▉                     | 0.012%
/model.13/cv1/conv/Conv:                      | ▉                     | 0.012%
/model.6/cv2/conv/Conv:                       | ▉                     | 0.011%
/model.13/m.0/cv1/conv/Conv:                  | ▉                     | 0.011%
/model.8/cv1/conv/Conv:                       | ▉                     | 0.010%
/model.13/m.0/cv2/conv/Conv:                  | ▉                     | 0.010%
/model.5/conv/Conv:                           | ▉                     | 0.010%
/model.6/m.0/m/m.1/cv1/conv/Conv:             | ▉                     | 0.009%
/model.23/cv3.0/cv3.0.0/cv3.0.0.1/conv/Conv:  | ▉                     | 0.008%
/model.23/cv2.2/cv2.2.0/conv/Conv:            | ▉                     | 0.008%
/model.23/cv2.1/cv2.1.1/conv/Conv:            | ▉                     | 0.008%
/model.19/m.0/cv2/conv/Conv:                  | ▉                     | 0.008%
/model.8/cv2/conv/Conv:                       | ▉                     | 0.008%
/model.9/cv1/conv/Conv:                       | ▉                     | 0.008%
/model.23/cv2.0/cv2.0.1/conv/Conv:            | ▉                     | 0.007%
/model.16/m.0/cv1/conv/Conv:                  | ▉                     | 0.007%
/model.17/conv/Conv:                          | ▉                     | 0.007%
/model.23/cv3.1/cv3.1.1/cv3.1.1.0/conv/Conv:  | ▉                     | 0.007%
/model.10/m/m.0/ffn/ffn.1/conv/Conv:          | ▉                     | 0.007%
/model.22/m.0/cv1/conv/Conv:                  |                       | 0.006%
/model.10/cv2/conv/Conv:                      |                       | 0.006%
/model.23/cv2.0/cv2.0.2/Conv:                 |                       | 0.006%
```

(continues on next page)

(continued from previous page)

```
/model.23/cv2.2/cv2.2.2/Conv:                 |                      | 0.005%
/model.23/cv2.1/cv2.1.2/Conv:                 |                      | 0.005%
/model.22/m.0/cv3/conv/Conv:                  |                      | 0.005%
/model.23/cv3.1/cv3.1.0/cv3.1.0.1/conv/Conv: |                      | 0.005%
/model.22/cv2/conv/Conv:                      |                      | 0.005%
/model.7/conv/Conv:                           |                      | 0.004%
/model.6/m.0/cv3/conv/Conv:                   |                      | 0.004%
/model.10/m/m.0/ffn/ffn.0/conv/Conv:          |                      | 0.004%
/model.8/m.0/m/m.1/cv2/conv/Conv:             |                      | 0.004%
/model.22/m.0/m/m.1/cv1/conv/Conv:            |                      | 0.004%
/model.8/m.0/m/m.1/cv1/conv/Conv:             |                      | 0.004%
/model.23/cv3.1/cv3.1.1/cv3.1.1.1/conv/Conv: |                      | 0.003%
/model.8/m.0/cv1/conv/Conv:                   |                      | 0.003%
/model.10/m/m.0/attn/proj/conv/Conv:          |                      | 0.003%
/model.22/m.0/m/m.0/cv2/conv/Conv:            |                      | 0.003%
PPQ_Operation_2:                              |                      | 0.003%
/model.8/m.0/cv3/conv/Conv:                   |                      | 0.003%
/model.6/m.0/m/m.0/cv1/conv/Conv:             |                      | 0.003%
/model.23/cv3.2/cv3.2.1/cv3.2.1.0/conv/Conv: |                      | 0.002%
/model.6/m.0/m/m.1/cv2/conv/Conv:             |                      | 0.002%
/model.8/m.0/m/m.0/cv2/conv/Conv:             |                      | 0.002%
/model.23/cv3.0/cv3.0.0/cv3.0.0.0/conv/Conv: |                      | 0.002%
/model.23/cv3.2/cv3.2.1/cv3.2.1.1/conv/Conv: |                      | 0.002%
/model.10/m/m.0/attn/MatMul_1:                |                      | 0.002%
/model.22/m.0/m/m.1/cv2/conv/Conv:            |                      | 0.001%
/model.6/m.0/m/m.0/cv2/conv/Conv:             |                      | 0.001%
/model.8/m.0/m/m.0/cv1/conv/Conv:             |                      | 0.001%
/model.23/cv3.0/cv3.0.1/cv3.0.1.0/conv/Conv: |                      | 0.001%
/model.23/cv3.2/cv3.2.0/cv3.2.0.1/conv/Conv: |                      | 0.001%
/model.2/cv2/conv/Conv:                       |                      | 0.001%
/model.23/cv3.0/cv3.0.1/cv3.0.1.1/conv/Conv: |                      | 0.001%
/model.6/m.0/cv1/conv/Conv:                   |                      | 0.001%
/model.23/cv3.2/cv3.2.2/Conv:                 |                      | 0.001%
/model.20/conv/Conv:                          |                      | 0.001%
/model.23/cv3.1/cv3.1.2/Conv:                 |                      | 0.001%
/model.23/cv3.2/cv3.2.0/cv3.2.0.0/conv/Conv: |                      | 0.001%
/model.6/m.0/cv2/conv/Conv:                   |                      | 0.001%
/model.23/cv3.0/cv3.0.2/Conv:                 |                      | 0.000%
/model.23/cv3.1/cv3.1.0/cv3.1.0.0/conv/Conv: |                      | 0.000%
/model.8/m.0/cv2/conv/Conv:                   |                      | 0.000%
/model.22/m.0/cv2/conv/Conv:                  |                      | 0.000%
/model.3/conv/Conv:                           |                      | 0.000%
/model.4/cv2/conv/Conv:                       |                      | 0.000%
```

Quantization error analysis

After using 16-bits quantization on layers with higher layer-wise error and employing horizontal layer split pass, the quantized model's mAP50:95 on COCO val2017 improves to 33.4% with the same inputs. Additionally, a noticeable decrease in cumulative error of output layers can be observed.

The graphwise error for the output layers of the model, /model.23/cv3.2/cv3.2.2/Conv, /model.23/cv2.2/cv2.2.2/Conv, /model.23/cv3.1/cv3.1.2/Conv, /model.23/cv2.1/cv2.1.2/Conv, /model.23/cv3.0/cv3.0.2/Conv and /model.23/cv2.0/cv2.0.2/Conv, are 0.771%, 8.156%, 0.339%, 4.222%, 0.190% and 4.232% respectively.

Quantization-Aware Training

To further improve the accuracy of the quantized model, we adopt the quantization-aware training(QAT) strategy. Here, QAT is performed based on 8-bit quantization.

Quantization settings

- yolo11n_qat.py
- trainer.py

Quantization results

```
Layer                                          | NOISE:SIGNAL POWER RATIO
/model.10/m/m.0/ffn/ffn.1/conv/Conv:           | ████████████████████ | 29.837%
/model.10/m/m.0/attn/proj/conv/Conv:           | ████████████████     | 23.397%
/model.10/m/m.0/attn/pe/conv/Conv:             | ██████████           | 15.253%
/model.23/cv3.1/cv3.1.1/cv3.1.1.1/conv/Conv:   | ██████████           | 14.819%
/model.10/m/m.0/attn/MatMul_1:                 | ██████████           | 14.725%
/model.23/cv3.0/cv3.0.1/cv3.0.1.1/conv/Conv:   | ██████████           | 14.315%
/model.23/cv3.2/cv3.2.0/cv3.2.0.1/conv/Conv:   | █████████            | 14.212%
/model.23/cv3.2/cv3.2.1/cv3.2.1.1/conv/Conv:   | █████████            | 14.187%
/model.10/m/m.0/attn/qkv/conv/Conv:            | █████████            | 13.797%
/model.23/cv2.2/cv2.2.0/conv/Conv:             | █████████            | 13.721%
/model.22/m.0/cv2/conv/Conv:                   | █████████            | 13.540%
/model.23/cv3.2/cv3.2.0/cv3.2.0.0/conv/Conv:   | █████████            | 13.408%
/model.8/m.0/cv2/conv/Conv:                    | █████████            | 12.809%
/model.22/m.0/cv3/conv/Conv:                   | ████████             | 12.623%
/model.23/cv2.1/cv2.1.1/conv/Conv:             | ████████             | 12.472%
/model.23/cv2.1/cv2.1.0/conv/Conv:             | ████████             | 12.177%
/model.22/m.0/m/m.1/cv2/conv/Conv:             | ████████             | 11.719%
/model.23/cv2.2/cv2.2.1/conv/Conv:             | ████████             | 11.711%
/model.10/cv1/conv/Conv:                       | ████████             | 11.589%
/model.22/cv2/conv/Conv:                       | ████████             | 11.551%
/model.23/cv2.0/cv2.0.1/conv/Conv:             | ████████             | 11.505%
/model.10/m/m.0/attn/MatMul:                   | ████████             | 11.346%
/model.22/cv1/conv/Conv:                       | ███████              | 10.201%
/model.23/cv3.1/cv3.1.0/cv3.1.0.1/conv/Conv:   | ██████               | 9.710%
/model.13/m.0/cv2/conv/Conv:                   | ██████               | 9.538%
/model.20/conv/Conv:                           | ██████               | 8.870%
/model.19/m.0/cv2/conv/Conv:                   | ██████               | 8.713%
/model.23/cv3.0/cv3.0.0/cv3.0.0.1/conv/Conv:   | █████                | 8.157%
/model.22/m.0/m/m.0/cv2/conv/Conv:             | █████                | 8.005%
/model.8/cv2/conv/Conv:                        | █████                | 7.952%
/model.8/m.0/cv1/conv/Conv:                    | █████                | 7.697%
/model.13/cv2/conv/Conv:                       | █████                | 7.557%
/model.19/cv2/conv/Conv:                       | █████                | 7.443%
/model.10/cv2/conv/Conv:                       | █████                | 7.403%
/model.6/m.0/cv2/conv/Conv:                    | █████                | 7.099%
/model.8/cv1/conv/Conv:                        | █████                | 6.996%
/model.19/cv1/conv/Conv:                       | █████                | 6.912%
/model.8/m.0/m/m.0/cv1/conv/Conv:              | ██████               | 6.908%
/model.8/m.0/cv3/conv/Conv:                    | ████                 | 6.755%
/model.23/cv3.2/cv3.2.1/cv3.2.1.0/conv/Conv:   | ████                 | 6.746%
/model.8/m.0/m/m.0/cv2/conv/Conv:              | ████                 | 6.743%
/model.8/m.0/m/m.1/cv1/conv/Conv:              | ████                 | 6.638%
/model.13/cv1/conv/Conv:                       | ████                 | 6.361%
/model.2/m.0/cv2/conv/Conv:                    | ████                 | 6.274%
/model.13/m.0/cv1/conv/Conv:                   | ████                 | 6.261%
```

(continues on next page)

(continued from previous page)

```
/model.19/m.0/cv1/conv/Conv:                  | ████                  | 6.191%
/model.22/m.0/m/m.0/cv1/conv/Conv:            | ████                  | 6.036%
/model.23/cv2.2/cv2.2.2/Conv:                 | ████                  | 5.999%
/model.22/m.0/m/m.1/cv1/conv/Conv:            | ████                  | 5.899%
/model.23/cv2.0/cv2.0.0/conv/Conv:            | ████                  | 5.618%
/model.8/m.0/m/m.1/cv2/conv/Conv:             | ████                  | 5.560%
/model.22/m.0/cv1/conv/Conv:                  | ███                   | 5.336%
/model.16/m.0/cv2/conv/Conv:                  | ███                   | 5.316%
/model.17/conv/Conv:                          | ███                   | 5.113%
/model.6/m.0/cv3/conv/Conv:                   | ███                   | 5.103%
/model.16/m.0/cv1/conv/Conv:                  | ███                   | 5.101%
/model.23/cv3.1/cv3.1.1/cv3.1.1.0/conv/Conv:  | ███                   | 5.052%
/model.2/cv2/conv/Conv:                       | ███                   | 5.003%
/model.6/cv2/conv/Conv:                       | ███                   | 4.968%
/model.6/cv1/conv/Conv:                       | ███                   | 4.792%
/model.23/cv2.1/cv2.1.2/Conv:                 | ███                   | 4.543%
/model.7/conv/Conv:                           | ███                   | 4.520%
/model.3/conv/Conv:                           | ███                   | 4.362%
/model.16/cv2/conv/Conv:                      | ███                   | 4.028%
/model.23/cv2.0/cv2.0.2/Conv:                 | ███                   | 4.001%
/model.23/cv3.1/cv3.1.0/cv3.1.0.0/conv/Conv:  | ███                   | 3.954%
/model.9/cv2/conv/Conv:                       | ███                   | 3.901%
/model.6/m.0/m/m.0/cv1/conv/Conv:             | ███                   | 3.891%
/model.10/m/m.0/ffn/ffn.0/conv/Conv:          | ██                    | 3.791%
/model.23/cv3.0/cv3.0.1/cv3.0.1.0/conv/Conv:  | ██                    | 3.711%
/model.4/cv1/conv/Conv:                       | ██                    | 3.673%
/model.6/m.0/m/m.0/cv2/conv/Conv:             | ██                    | 3.620%
/model.6/m.0/m/m.1/cv1/conv/Conv:             | ██                    | 3.513%
/model.4/cv2/conv/Conv:                       | ██                    | 3.421%
/model.5/conv/Conv:                           | ██                    | 3.320%
/model.6/m.0/cv1/conv/Conv:                   | ██                    | 3.073%
/model.2/cv1/conv/Conv:                       | ██                    | 3.021%
/model.16/cv1/conv/Conv:                      | ██                    | 2.764%
/model.6/m.0/m/m.1/cv2/conv/Conv:             | ██                    | 2.454%
/model.4/m.0/cv1/conv/Conv:                   | ██                    | 2.408%
/model.4/m.0/cv2/conv/Conv:                   | ▌                     | 1.689%
/model.2/m.0/cv1/conv/Conv:                   | ▌                     | 1.602%
/model.9/cv1/conv/Conv:                       | ▌                     | 1.568%
/model.1/conv/Conv:                           | ▌                     | 1.205%
/model.23/cv3.0/cv3.0.0/cv3.0.0.0/conv/Conv:  | ▌                     | 1.091%
/model.23/cv3.2/cv3.2.2/Conv:                 |                       | 0.746%
/model.23/cv3.1/cv3.1.2/Conv:                 |                       | 0.480%
/model.23/cv3.0/cv3.0.2/Conv:                 |                       | 0.386%
/model.0/conv/Conv:                           |                       | 0.163%
Analysing Layerwise quantization error:: 100%|██████████| 89/89 [04:01<00:00,  2.72s/
↪it]
Layer                                         | NOISE:SIGNAL POWER RATIO
/model.2/cv2/conv/Conv:                       | ████████████████████  | 0.935%
/model.9/cv2/conv/Conv:                       | ██████████████████    | 0.826%
/model.2/m.0/cv1/conv/Conv:                   | ███████████████       | 0.698%
/model.3/conv/Conv:                           | ████████████          | 0.611%
/model.4/cv2/conv/Conv:                       | ██████████            | 0.491%
/model.10/cv2/conv/Conv:                      | ██████████            | 0.408%
/model.23/cv2.2/cv2.2.2/Conv:                 | ██████                | 0.283%
/model.2/cv1/conv/Conv:                       | ██████                | 0.261%
/model.4/cv1/conv/Conv:                       | ▌                     | 0.249%
/model.1/conv/Conv:                           | ▌                     | 0.217%
```

(continues on next page)

(continued from previous page)

```
/model.22/cv1/conv/Conv:                    | ████                | 0.201%
/model.10/cv1/conv/Conv:                    | ███                 | 0.143%
/model.5/conv/Conv:                         | ███                 | 0.136%
/model.16/cv1/conv/Conv:                    | ███                 | 0.128%
/model.10/m/m.0/attn/pe/conv/Conv:          | ███                 | 0.120%
/model.0/conv/Conv:                         | ███                 | 0.118%
/model.16/m.0/cv1/conv/Conv:                | ██                  | 0.105%
/model.16/cv2/conv/Conv:                    | ██                  | 0.094%
/model.16/m.0/cv2/conv/Conv:                | ██                  | 0.092%
/model.23/cv2.0/cv2.0.0/conv/Conv:          | ██                  | 0.089%
/model.4/m.0/cv1/conv/Conv:                 | ██                  | 0.071%
/model.22/m.0/cv1/conv/Conv:                | █                   | 0.067%
/model.19/cv2/conv/Conv:                    | █                   | 0.063%
/model.6/cv2/conv/Conv:                     | █                   | 0.061%
/model.4/m.0/cv2/conv/Conv:                 | █                   | 0.059%
/model.17/conv/Conv:                        | █                   | 0.054%
/model.13/cv2/conv/Conv:                    | █                   | 0.053%
/model.8/m.0/cv3/conv/Conv:                 | █                   | 0.051%
/model.6/cv1/conv/Conv:                     | █                   | 0.047%
/model.23/cv2.2/cv2.2.0/conv/Conv:          | █                   | 0.042%
/model.23/cv3.0/cv3.0.0/cv3.0.0.1/conv/Conv: | █                  | 0.041%
/model.13/cv1/conv/Conv:                    | █                   | 0.040%
/model.7/conv/Conv:                         | █                   | 0.038%
/model.10/m/m.0/attn/qkv/conv/Conv:         | █                   | 0.038%
/model.13/m.0/cv1/conv/Conv:                | █                   | 0.033%
/model.23/cv2.1/cv2.1.0/conv/Conv:          | █                   | 0.031%
/model.6/m.0/m/m.1/cv1/conv/Conv:           | █                   | 0.028%
/model.19/m.0/cv2/conv/Conv:                | █                   | 0.027%
/model.8/m.0/m/m.1/cv1/conv/Conv:           | █                   | 0.026%
/model.2/m.0/cv2/conv/Conv:                 | █                   | 0.026%
/model.19/m.0/cv1/conv/Conv:                |                     | 0.022%
/model.6/m.0/cv3/conv/Conv:                 |                     | 0.021%
/model.19/cv1/conv/Conv:                    |                     | 0.021%
/model.9/cv1/conv/Conv:                     |                     | 0.016%
/model.22/m.0/m/m.1/cv1/conv/Conv:          |                     | 0.016%
/model.13/m.0/cv2/conv/Conv:                |                     | 0.015%
/model.23/cv3.1/cv3.1.0/cv3.1.0.1/conv/Conv: |                    | 0.015%
/model.22/m.0/m/m.0/cv1/conv/Conv:          |                     | 0.014%
/model.8/cv1/conv/Conv:                     |                     | 0.013%
/model.23/cv2.0/cv2.0.2/Conv:               |                     | 0.013%
/model.23/cv2.2/cv2.2.1/conv/Conv:          |                     | 0.012%
/model.10/m/m.0/ffn/ffn.0/conv/Conv:        |                     | 0.011%
/model.23/cv3.2/cv3.2.0/cv3.2.0.1/conv/Conv: |                    | 0.011%
/model.8/cv2/conv/Conv:                     |                     | 0.011%
/model.23/cv2.1/cv2.1.2/Conv:               |                     | 0.010%
/model.22/m.0/cv3/conv/Conv:                |                     | 0.010%
/model.23/cv2.1/cv2.1.1/conv/Conv:          |                     | 0.008%
/model.10/m/m.0/ffn/ffn.1/conv/Conv:        |                     | 0.008%
/model.23/cv2.0/cv2.0.1/conv/Conv:          |                     | 0.007%
/model.10/m/m.0/attn/proj/conv/Conv:        |                     | 0.007%
/model.8/m.0/cv1/conv/Conv:                 |                     | 0.007%
/model.22/m.0/m/m.0/cv2/conv/Conv:          |                     | 0.006%
/model.8/m.0/m/m.1/cv2/conv/Conv:           |                     | 0.005%
/model.22/cv2/conv/Conv:                    |                     | 0.005%
/model.20/conv/Conv:                        |                     | 0.005%
/model.23/cv3.1/cv3.1.1/cv3.1.1.0/conv/Conv: |                    | 0.005%
/model.6/m.0/m/m.0/cv1/conv/Conv:           |                     | 0.005%
```

(continues on next page)

(continued from previous page)

```
/model.8/m.0/m/m.0/cv1/conv/Conv:             |                       | 0.004%
/model.23/cv3.1/cv3.1.1/cv3.1.1.1/conv/Conv: |                       | 0.003%
/model.8/m.0/m/m.0/cv2/conv/Conv:             |                       | 0.003%
/model.23/cv3.0/cv3.0.0/cv3.0.0.0/conv/Conv: |                       | 0.003%
/model.6/m.0/cv1/conv/Conv:                   |                       | 0.003%
/model.23/cv3.2/cv3.2.2/Conv:                 |                       | 0.003%
/model.23/cv3.2/cv3.2.1/cv3.2.1.0/conv/Conv: |                       | 0.003%
/model.6/m.0/m/m.1/cv2/conv/Conv:             |                       | 0.003%
/model.23/cv3.2/cv3.2.1/cv3.2.1.1/conv/Conv: |                       | 0.002%
/model.22/m.0/m/m.1/cv2/conv/Conv:            |                       | 0.002%
/model.6/m.0/m/m.0/cv2/conv/Conv:             |                       | 0.002%
/model.23/cv3.0/cv3.0.1/cv3.0.1.0/conv/Conv: |                       | 0.002%
/model.10/m/m.0/attn/MatMul_1:                |                       | 0.002%
/model.23/cv3.0/cv3.0.2/Conv:                 |                       | 0.001%
/model.23/cv3.1/cv3.1.2/Conv:                 |                       | 0.001%
/model.23/cv3.0/cv3.0.1/cv3.0.1.1/conv/Conv: |                       | 0.001%
/model.23/cv3.1/cv3.1.0/cv3.1.0.0/conv/Conv: |                       | 0.001%
/model.23/cv3.2/cv3.2.0/cv3.2.0.0/conv/Conv: |                       | 0.001%
/model.6/m.0/cv2/conv/Conv:                   |                       | 0.000%
/model.10/m/m.0/attn/MatMul:                  |                       | 0.000%
/model.8/m.0/cv2/conv/Conv:                   |                       | 0.000%
/model.22/m.0/cv2/conv/Conv:                  |                       | 0.000%
```

Quantization error analysis

After applying QAT to 8-bit quantization, the quantized model's mAP50:95 on COCO val2017 improves to 36.0% with the same inputs, while cumulative errors of out layers are significantly reduced. Compared to the other two quantization methods, the 8-bit QAT quantized model achieves the highest quantization accuracy with the lowest inference latency.

The graphwise error for the output layers of the model, /model.23/cv3.2/cv3.2.2/Conv, /model.23/cv2.2/cv2.2.2/Conv, /model.23/cv3.1/cv3.1.2/Conv, /model.23/cv2.1/cv2.1.2/Conv, /model.23/cv3.0/cv3.0.2/Conv and /model.23/cv2.0/cv2.0.2/Conv, are 0.746%, 5.999%, 0.480%, 4.543%, 0.386% and 4.001% respectively.

Note: If the model inference speed is a higher priority and a certain degree of accuracy loss is acceptable, you may consider quantizing the model with an input size of 320x320 for the YOLO11N model. The model inference speed of different input resolutions can be found in README.md .

3.7.3 Model deployment

example

Object detection base class

- dl_detect_base.hpp
- dl_detect_base.cpp

Pre-process

`ImagePreprocessor` class contains the common pre-precoess pipeline, `color conversion`, `crop`, `resize`, `normalization`, `quantize`

- dl_image_preprocessor.hpp
- dl_image_preprocessor.cpp

Post-process

- dl_detect_postprocessor.hpp
- dl_detect_postprocessor.cpp
- dl_detect_yolo11_postprocessor.hpp
- dl_detect_yolo11_postprocessor.cpp

3.8 How to deploy YOLO11n-pose

In this tutorial, we will introduce how to quantize a pre-trained YOLO11n-pose model using ESP-PPQ and deploy the quantized YOLO11n-pose model using ESP-DL.

3.8.1 Preparation

1. *ESP_IDF*
2. *ESP_PPQ*

3.8.2 Model quantization

Pre-trained Model

You can download pre-trained yolo11n-pose model from Ultralytics release.

Currently, ESP-PPQ supports ONNX, PyTorch, and TensorFlow models. During the quantization process, PyTorch and TensorFlow models are first converted to ONNX models, so the pre-trained yolo11n-pose model needs to be converted to an ONNX model.

Specifically, refer to the script export_onnx.py to convert the pre-trained yolo11n-pose model to an ONNX model.

In the script, we have overridden the forward method of the Pose class, which offers following advantages:

- Faster inference. Compared to the original yolo11n-pose model, operations related to decoding bounding boxes and keypoints in Pose head are moved from the inference pass to the post-processing phase, resulting in a significant reduction in inference latency. On one hand, operations like `Conv`, `Transpose`, `Slice`, `Split` and `Concat` are time-consuming when applied during inference pass. On the other hand, the inference outputs are first filtered using a score threshold before decoding the boxes in the post-processing pass, which significantly reduces the number of calculations, thereby accelerating the overall inference speed.
- Lower quantization Error. The `Concat` and `Add` operators adopt joint quantization in ESP-PPQ. To reduce quantization errors, the box and score are output by separate branches, rather than being concatenated, due to the significant difference in their ranges. Similarly, since the ranges of the two inputs of `Add` and `Sub` differ significantly, the calculations are performed in the post-processing phase to avoid quantization errors.

Calibration Dataset

The calibration dataset needs to match the input format of the model. The calibration dataset should cover all possible input scenarios to better quantize the model. Here, the calibration dataset used in this example is calib_yolo11n-pose.

8bit default configuration quantization

Quantization settings

```
target="esp32p4"
num_of_bits=8
batch_size=32
quant_setting = QuantizationSettingFactory.espdl_setting() # default setting
```

Quantization results

```
Layer                                        | NOISE:SIGNAL POWER RATIO
/model.22/m.0/cv2/conv/Conv:                 | ████████████████████ | 29.305%
/model.23/cv3.2/cv3.2.0/cv3.2.0.1/conv/Conv: | ██████████████████   | 26.959%
/model.23/cv4.1/cv4.1.0/conv/Conv:           | ██████████████████   | 26.555%
/model.23/cv3.2/cv3.2.1/cv3.2.1.0/conv/Conv: | ████████████████     | 25.611%
/model.20/conv/Conv:                         | ████████████████     | 24.738%
/model.23/cv3.2/cv3.2.0/cv3.2.0.0/conv/Conv: | ███████████████      | 24.122%
/model.23/cv4.1/cv4.1.1/conv/Conv:           | ██████████████       | 22.512%
/model.19/m.0/cv2/conv/Conv:                 | ██████████████       | 22.397%
/model.23/cv2.0/cv2.0.1/conv/Conv:           | ██████████████       | 22.174%
/model.23/cv4.0/cv4.0.0/conv/Conv:           | ██████████████       | 21.621%
/model.23/cv2.1/cv2.1.1/conv/Conv:           | ██████████████       | 21.489%
/model.23/cv4.0/cv4.0.1/conv/Conv:           | ██████████████       | 21.445%
/model.23/cv3.1/cv3.1.0/cv3.1.0.1/conv/Conv: | ████████████         | 20.528%
```

(continues on next page)

(continued from previous page)

```
/model.23/cv3.1/cv3.1.1/cv3.1.1.0/conv/Conv: | ████████████ | 20.083%
/model.23/cv3.1/cv3.1.0/cv3.1.0.0/conv/Conv: | ████████████ | 20.066%
/model.13/m.0/cv2/conv/Conv:                 | ████████████ | 20.042%
/model.22/m.0/cv3/conv/Conv:                 | ███████████  | 19.737%
/model.10/m/m.0/ffn/ffn.1/conv/Conv:         | ███████████  | 19.585%
/model.23/cv3.1/cv3.1.1/cv3.1.1.1/conv/Conv: | ███████████  | 19.392%
/model.23/cv3.0/cv3.0.1/cv3.0.1.0/conv/Conv: | ███████████  | 18.773%
/model.23/cv3.2/cv3.2.1/cv3.2.1.1/conv/Conv: | ███████████  | 18.688%
/model.22/cv1/conv/Conv:                     | ███████████  | 18.579%
/model.19/cv2/conv/Conv:                     | ███████████  | 18.494%
/model.22/m.0/m/m.1/cv2/conv/Conv:           | ██████████   | 17.576%
/model.17/conv/Conv:                         | ██████████   | 17.224%
/model.19/cv1/conv/Conv:                     | ██████████   | 17.140%
/model.22/cv2/conv/Conv:                     | ██████████   | 16.785%
/model.23/cv4.2/cv4.2.1/conv/Conv:           | ██████████   | 16.375%
/model.23/cv4.2/cv4.2.0/conv/Conv:           | ██████████   | 16.167%
/model.23/cv2.1/cv2.1.0/conv/Conv:           | ██████████   | 15.655%
/model.23/cv3.0/cv3.0.1/cv3.0.1.1/conv/Conv: | ██████████   | 15.504%
/model.23/cv2.2/cv2.2.0/conv/Conv:           | ██████████   | 15.431%
/model.10/m/m.0/attn/proj/conv/Conv:         | █████████    | 15.251%
/model.23/cv3.0/cv3.0.0/cv3.0.0.1/conv/Conv: | █████████    | 15.171%
/model.22/m.0/m/m.0/cv2/conv/Conv:           | █████████    | 15.006%
/model.19/m.0/cv1/conv/Conv:                 | █████████    | 14.692%
/model.23/cv2.2/cv2.2.1/conv/Conv:           | █████████    | 14.548%
/model.22/m.0/m/m.0/cv1/conv/Conv:           | ████████     | 13.065%
/model.16/m.0/cv2/conv/Conv:                 | ████████     | 12.980%
/model.22/m.0/m/m.1/cv1/conv/Conv:           | ████████     | 12.921%
/model.10/m/m.0/attn/pe/conv/Conv:           | ████████     | 12.745%
/model.23/cv4.1/cv4.1.2/Conv:                | ███████      | 12.498%
/model.13/cv2/conv/Conv:                     | ███████      | 11.932%
/model.23/cv4.2/cv4.2.2/Conv:                | ███████      | 11.797%
/model.13/m.0/cv1/conv/Conv:                 | ███████      | 11.777%
/model.16/cv2/conv/Conv:                     | ██████       | 10.892%
/model.13/cv1/conv/Conv:                     | ██████       | 10.760%
/model.23/cv2.0/cv2.0.0/conv/Conv:           | ██████       | 10.352%
/model.23/cv4.0/cv4.0.2/Conv:                | ██████       | 10.325%
/model.22/m.0/cv1/conv/Conv:                 | ██████       | 10.257%
/model.8/m.0/cv2/conv/Conv:                  | ██████       | 9.687%
/model.10/m/m.0/ffn/ffn.0/conv/Conv:         | ██████       | 8.997%
/model.10/cv1/conv/Conv:                     | ██████       | 8.787%
/model.16/m.0/cv1/conv/Conv:                 | ██████       | 8.629%
/model.10/m/m.0/attn/qkv/conv/Conv:          | ██████       | 8.600%
/model.8/m.0/cv3/conv/Conv:                  | ██████       | 8.328%
/model.10/m/m.0/attn/MatMul_1:               | ██████       | 8.293%
/model.16/cv1/conv/Conv:                     | ██████       | 7.947%
/model.10/cv2/conv/Conv:                     | ██████       | 7.824%
/model.8/cv2/conv/Conv:                      | ██████       | 7.696%
/model.23/cv3.0/cv3.0.0/cv3.0.0.0/conv/Conv: | ██████       | 7.615%
/model.8/m.0/m/m.1/cv2/conv/Conv:            | ██████       | 7.145%
/model.8/m.0/m/m.0/cv2/conv/Conv:            | ██████       | 7.033%
/model.10/m/m.0/attn/MatMul:                 | ██████       | 6.707%
/model.8/m.0/m/m.1/cv1/conv/Conv:            | ████         | 6.376%
/model.23/cv2.1/cv2.1.2/Conv:                | ████         | 6.321%
/model.8/cv1/conv/Conv:                      | ████         | 6.296%
/model.6/m.0/cv2/conv/Conv:                  | ████         | 5.605%
/model.23/cv3.2/cv3.2.2/Conv:                | ████         | 5.599%
/model.6/m.0/m/m.0/cv2/conv/Conv:            | ████         | 5.559%
```

(continues on next page)

(continued from previous page)

```
/model.23/cv2.0/cv2.0.2/Conv:                | ████                | 5.262%
/model.23/cv2.2/cv2.2.2/Conv:                | ████                | 5.207%
/model.6/m.0/cv3/conv/Conv:                  | ███                 | 4.840%
/model.8/m.0/cv1/conv/Conv:                  | ███                 | 4.667%
/model.6/cv1/conv/Conv:                      | ███                 | 4.523%
/model.9/cv1/conv/Conv:                      | ███                 | 4.038%
/model.9/cv2/conv/Conv:                      | ██                  | 3.733%
/model.7/conv/Conv:                          | ██                  | 3.605%
/model.6/cv2/conv/Conv:                      | ██                  | 3.478%
/model.3/conv/Conv:                          | ██                  | 3.352%
/model.2/cv2/conv/Conv:                      | ██                  | 3.230%
/model.8/m.0/m/m.0/cv1/conv/Conv:            | ██                  | 3.136%
/model.4/cv1/conv/Conv:                      | ██                  | 2.913%
/model.6/m.0/m/m.1/cv1/conv/Conv:            | ██                  | 2.830%
/model.6/m.0/m/m.1/cv2/conv/Conv:            | ██                  | 2.692%
/model.6/m.0/m/m.0/cv1/conv/Conv:            | ██                  | 2.557%
/model.6/m.0/cv1/conv/Conv:                  | ██                  | 2.475%
/model.5/conv/Conv:                          | ██                  | 2.413%
/model.2/cv1/conv/Conv:                      | ▌                   | 2.267%
/model.4/cv2/conv/Conv:                      | ▌                   | 2.135%
/model.2/m.0/cv2/conv/Conv:                  | ▌                   | 2.104%
/model.4/m.0/cv1/conv/Conv:                  | ▌                   | 1.910%
/model.1/conv/Conv:                          | ▌                   | 1.708%
/model.2/m.0/cv1/conv/Conv:                  | ▌                   | 1.658%
/model.23/cv3.1/cv3.1.2/Conv:                | ▌                   | 1.455%
/model.4/m.0/cv2/conv/Conv:                  | ▌                   | 1.056%
/model.23/cv3.0/cv3.0.2/Conv:                |                     | 0.364%
/model.0/conv/Conv:                          |                     | 0.087%
Analysing Layerwise quantization error:: 100%|██████████| 98/98 [16:09<00:00,  9.89s/
↪it]
Layer                                        | NOISE:SIGNAL POWER RATIO
/model.0/conv/Conv:                          | ████████████████████ | 1.017%
/model.9/cv2/conv/Conv:                      | ██████████           | 0.493%
/model.8/cv1/conv/Conv:                      | ████████             | 0.410%
/model.2/cv2/conv/Conv:                      | ██████               | 0.287%
/model.1/conv/Conv:                          | ████                 | 0.228%
/model.2/cv1/conv/Conv:                      | ███                  | 0.163%
/model.16/cv2/conv/Conv:                     | ███                  | 0.130%
/model.4/cv2/conv/Conv:                      | ██                   | 0.096%
/model.3/conv/Conv:                          | ▌                    | 0.070%
/model.4/cv1/conv/Conv:                      | ▌                    | 0.068%
/model.10/cv1/conv/Conv:                     | ▌                    | 0.049%
/model.2/m.0/cv2/conv/Conv:                  | ▌                    | 0.047%
/model.2/m.0/cv1/conv/Conv:                  | ▌                    | 0.043%
/model.4/m.0/cv2/conv/Conv:                  | ▌                    | 0.041%
/model.13/cv2/conv/Conv:                     | ▌                    | 0.037%
/model.16/cv1/conv/Conv:                     | ▌                    | 0.030%
/model.22/cv2/conv/Conv:                     | ▌                    | 0.027%
/model.8/cv2/conv/Conv:                      | ▌                    | 0.027%
/model.13/cv1/conv/Conv:                     |                      | 0.025%
/model.5/conv/Conv:                          |                      | 0.025%
/model.19/m.0/cv2/conv/Conv:                 |                      | 0.025%
/model.6/cv2/conv/Conv:                      |                      | 0.024%
/model.4/m.0/cv1/conv/Conv:                  |                      | 0.022%
/model.6/cv1/conv/Conv:                      |                      | 0.021%
/model.19/cv1/conv/Conv:                     |                      | 0.020%
/model.23/cv2.1/cv2.1.1/conv/Conv:           |                      | 0.018%
```

(continues on next page)

(continued from previous page)

```
/model.23/cv4.1/cv4.1.0/conv/Conv:           |                       | 0.017%
/model.9/cv1/conv/Conv:                      |                       | 0.015%
/model.23/cv4.2/cv4.2.1/conv/Conv:           |                       | 0.014%
/model.10/m/m.0/attn/qkv/conv/Conv:          |                       | 0.014%
/model.19/cv2/conv/Conv:                     |                       | 0.014%
/model.16/m.0/cv2/conv/Conv:                 |                       | 0.014%
/model.23/cv4.2/cv4.2.0/conv/Conv:           |                       | 0.014%
/model.6/m.0/m/m.0/cv1/conv/Conv:            |                       | 0.013%
/model.22/m.0/cv3/conv/Conv:                 |                       | 0.013%
/model.23/cv3.2/cv3.2.0/cv3.2.0.1/conv/Conv: |                       | 0.013%
/model.23/cv4.0/cv4.0.0/conv/Conv:           |                       | 0.013%
/model.23/cv3.1/cv3.1.1/cv3.1.1.1/conv/Conv: |                       | 0.013%
/model.22/m.0/m/m.1/cv1/conv/Conv:           |                       | 0.012%
/model.6/m.0/cv3/conv/Conv:                  |                       | 0.012%
/model.10/m/m.0/attn/pe/conv/Conv:           |                       | 0.012%
/model.23/cv4.1/cv4.1.1/conv/Conv:           |                       | 0.011%
/model.8/m.0/m/m.1/cv1/conv/Conv:            |                       | 0.011%
/model.13/m.0/cv1/conv/Conv:                 |                       | 0.011%
/model.22/m.0/m/m.0/cv1/conv/Conv:           |                       | 0.011%
/model.6/m.0/m/m.1/cv1/conv/Conv:            |                       | 0.011%
/model.23/cv3.2/cv3.2.1/cv3.2.1.1/conv/Conv: |                       | 0.011%
/model.8/m.0/cv3/conv/Conv:                  |                       | 0.010%
/model.7/conv/Conv:                          |                       | 0.010%
/model.17/conv/Conv:                         |                       | 0.009%
/model.8/m.0/m/m.0/cv1/conv/Conv:            |                       | 0.009%
/model.13/m.0/cv2/conv/Conv:                 |                       | 0.009%
/model.10/m/m.0/attn/MatMul:                 |                       | 0.009%
/model.19/m.0/cv1/conv/Conv:                 |                       | 0.008%
/model.16/m.0/cv1/conv/Conv:                 |                       | 0.008%
/model.23/cv2.2/cv2.2.1/conv/Conv:           |                       | 0.008%
/model.8/m.0/m/m.1/cv2/conv/Conv:            |                       | 0.008%
/model.8/m.0/cv1/conv/Conv:                  |                       | 0.008%
/model.10/cv2/conv/Conv:                     |                       | 0.007%
/model.23/cv2.0/cv2.0.2/Conv:                |                       | 0.007%
/model.22/m.0/cv1/conv/Conv:                 |                       | 0.007%
/model.6/m.0/cv1/conv/Conv:                  |                       | 0.007%
/model.23/cv2.0/cv2.0.0/conv/Conv:           |                       | 0.006%
/model.23/cv2.1/cv2.1.0/conv/Conv:           |                       | 0.006%
/model.22/m.0/m/m.1/cv2/conv/Conv:           |                       | 0.006%
/model.23/cv3.2/cv3.2.1/cv3.2.1.0/conv/Conv: |                       | 0.005%
/model.8/m.0/m/m.0/cv2/conv/Conv:            |                       | 0.005%
/model.23/cv2.1/cv2.1.2/Conv:                |                       | 0.005%
/model.23/cv3.2/cv3.2.0/cv3.2.0.0/conv/Conv: |                       | 0.005%
/model.23/cv2.2/cv2.2.2/Conv:                |                       | 0.005%
/model.22/cv1/conv/Conv:                     |                       | 0.004%
/model.10/m/m.0/attn/proj/conv/Conv:         |                       | 0.004%
/model.23/cv4.2/cv4.2.2/Conv:                |                       | 0.004%
/model.23/cv4.1/cv4.1.2/Conv:                |                       | 0.004%
/model.22/m.0/m/m.0/cv2/conv/Conv:           |                       | 0.004%
/model.23/cv2.2/cv2.2.0/conv/Conv:           |                       | 0.003%
/model.6/m.0/m/m.1/cv2/conv/Conv:            |                       | 0.003%
/model.23/cv4.0/cv4.0.1/conv/Conv:           |                       | 0.003%
/model.6/m.0/m/m.0/cv2/conv/Conv:            |                       | 0.003%
/model.10/m/m.0/attn/MatMul_1:               |                       | 0.002%
/model.23/cv4.0/cv4.0.2/Conv:                |                       | 0.002%
/model.10/m/m.0/ffn/ffn.1/conv/Conv:         |                       | 0.002%
/model.20/conv/Conv:                         |                       | 0.002%
```

(continues on next page)

(continued from previous page)

```
/model.23/cv2.0/cv2.0.1/conv/Conv:           |                        | 0.002%
/model.10/m/m.0/ffn/ffn.0/conv/Conv:         |                        | 0.001%
/model.23/cv3.1/cv3.1.0/cv3.1.0.1/conv/Conv: |                        | 0.001%
/model.23/cv3.1/cv3.1.1/cv3.1.1.0/conv/Conv: |                        | 0.001%
/model.23/cv3.2/cv3.2.2/Conv:                |                        | 0.001%
/model.23/cv3.0/cv3.0.1/cv3.0.1.0/conv/Conv: |                        | 0.001%
/model.23/cv3.1/cv3.1.0/cv3.1.0.0/conv/Conv: |                        | 0.001%
/model.23/cv3.1/cv3.1.2/Conv:                |                        | 0.000%
/model.23/cv3.0/cv3.0.2/Conv:                |                        | 0.000%
/model.23/cv3.0/cv3.0.0/cv3.0.0.1/conv/Conv: |                        | 0.000%
/model.6/m.0/cv2/conv/Conv:                  |                        | 0.000%
/model.23/cv3.0/cv3.0.1/cv3.0.1.1/conv/Conv: |                        | 0.000%
/model.23/cv3.0/cv3.0.0/cv3.0.0.0/conv/Conv: |                        | 0.000%
/model.8/m.0/cv2/conv/Conv:                  |                        | 0.000%
/model.22/m.0/cv2/conv/Conv:                 |                        | 0.000%
```

Quantization error analysis

With the same inputs, The Pose mAP50:95 on COCO after quantization is only 43.1%, which is lower than that of the float model（50.0%）.

Quantization-Aware Training

To further improve the accuracy of the quantized model, we adopt the quantization-aware training(QAT) strategy. Here, QAT is performed based on 8-bit quantization.

Quantization settings

- yolo11n-pose_qat.py
- trainer.py

Quantization results

```
Layer                                        | NOISE:SIGNAL POWER RATIO
/model.22/m.0/cv2/conv/Conv:                 | ████████████████████ | 27.739%
/model.23/cv3.2/cv3.2.0/cv3.2.0.1/conv/Conv: | ███████████████████  | 26.872%
/model.23/cv4.1/cv4.1.0/conv/Conv:           | ███████████████████  | 26.229%
/model.23/cv2.1/cv2.1.1/conv/Conv:           | ██████████████████   | 25.300%
/model.23/cv3.2/cv3.2.1/cv3.2.1.0/conv/Conv: | ██████████████████   | 24.625%
/model.23/cv2.0/cv2.0.1/conv/Conv:           | ████████████████     | 23.751%
/model.20/conv/Conv:                         | ████████████████     | 23.320%
/model.23/cv3.2/cv3.2.0/cv3.2.0.0/conv/Conv: | ████████████████     | 22.901%
/model.23/cv4.1/cv4.1.1/conv/Conv:           | ███████████████      | 22.516%
/model.10/m/m.0/ffn/ffn.1/conv/Conv:         | ███████████████      | 22.035%
/model.19/m.0/cv2/conv/Conv:                 | ███████████████      | 21.569%
/model.23/cv4.0/cv4.0.0/conv/Conv:           | ██████████████       | 21.199%
/model.23/cv3.1/cv3.1.0/cv3.1.0.1/conv/Conv: | ██████████████       | 20.785%
/model.23/cv3.1/cv3.1.1/cv3.1.1.0/conv/Conv: | ██████████████       | 20.597%
/model.23/cv3.1/cv3.1.1/cv3.1.1.1/conv/Conv: | ██████████████       | 20.329%
/model.23/cv4.0/cv4.0.1/conv/Conv:           | ██████████████       | 20.179%
/model.23/cv3.1/cv3.1.0/cv3.1.0.0/conv/Conv: | ████████████         | 19.983%
/model.22/m.0/cv3/conv/Conv:                 | ████████████         | 19.919%
/model.13/m.0/cv2/conv/Conv:                 | ████████████         | 19.424%
/model.23/cv3.0/cv3.0.1/cv3.0.1.0/conv/Conv: | ████████████         | 18.893%
/model.19/cv2/conv/Conv:                     | ███████████          | 18.055%
/model.23/cv3.2/cv3.2.1/cv3.2.1.1/conv/Conv: | ███████████          | 17.915%
```

(continues on next page)

(continued from previous page)

```
/model.22/m.0/m/m.1/cv2/conv/Conv:            | ██████████ | 17.796%
/model.22/cv1/conv/Conv:                      | ██████████ | 17.777%
/model.23/cv4.2/cv4.2.1/conv/Conv:            | ██████████ | 17.573%
/model.19/cv1/conv/Conv:                      | ██████████ | 17.116%
/model.17/conv/Conv:                          | ██████████ | 16.869%
/model.22/cv2/conv/Conv:                      | ██████████ | 16.750%
/model.23/cv2.2/cv2.2.1/conv/Conv:            | ██████████ | 16.540%
/model.10/m/m.0/attn/proj/conv/Conv:          | ██████████ | 16.491%
/model.23/cv2.2/cv2.2.0/conv/Conv:            | ██████████ | 16.421%
/model.23/cv2.1/cv2.1.0/conv/Conv:            | ██████████ | 16.205%
/model.23/cv4.2/cv4.2.0/conv/Conv:            | ██████████ | 16.116%
/model.23/cv3.0/cv3.0.1/cv3.0.1.1/conv/Conv:  | ██████████ | 15.400%
/model.22/m.0/m/m.0/cv2/conv/Conv:            | ██████████ | 15.251%
/model.23/cv3.0/cv3.0.0/cv3.0.0.1/conv/Conv:  | ██████████ | 14.851%
/model.10/m/m.0/attn/pe/conv/Conv:            | ██████████ | 14.659%
/model.19/m.0/cv1/conv/Conv:                  | ██████████ | 14.289%
/model.22/m.0/m/m.1/cv1/conv/Conv:            | ████████ | 13.038%
/model.16/m.0/cv2/conv/Conv:                  | ████████ | 12.941%
/model.22/m.0/m/m.0/cv1/conv/Conv:            | ████████ | 12.791%
/model.23/cv4.2/cv4.2.2/Conv:                 | ████████ | 12.508%
/model.23/cv4.1/cv4.1.2/Conv:                 | ████████ | 12.226%
/model.13/cv1/conv/Conv:                      | ████████ | 11.821%
/model.13/cv2/conv/Conv:                      | ████████ | 11.612%
/model.13/m.0/cv1/conv/Conv:                  | ████████ | 11.515%
/model.10/m/m.0/attn/MatMul_1:                | ████████ | 11.303%
/model.16/cv2/conv/Conv:                      | ████████ | 11.028%
/model.10/m/m.0/attn/qkv/conv/Conv:           | ████████ | 10.951%
/model.10/cv1/conv/Conv:                      | ████████ | 10.755%
/model.23/cv2.0/cv2.0.0/conv/Conv:            | ████████ | 10.684%
/model.22/m.0/cv1/conv/Conv:                  | ███████ | 10.164%
/model.10/m/m.0/ffn/ffn.0/conv/Conv:          | ███████ | 9.968%
/model.16/m.0/cv1/conv/Conv:                  | ███████ | 9.656%
/model.23/cv4.0/cv4.0.2/Conv:                 | ███████ | 9.566%
/model.8/m.0/cv2/conv/Conv:                   | ███████ | 9.521%
/model.10/cv2/conv/Conv:                      | ██████ | 8.068%
/model.16/cv1/conv/Conv:                      | ██████ | 7.989%
/model.23/cv2.1/cv2.1.2/Conv:                 | ██████ | 7.969%
/model.8/m.0/cv3/conv/Conv:                   | ██████ | 7.725%
/model.23/cv3.0/cv3.0.0/cv3.0.0.0/conv/Conv:  | █████ | 7.570%
/model.8/m.0/m/m.0/cv2/conv/Conv:             | █████ | 7.339%
/model.8/m.0/m/m.1/cv2/conv/Conv:             | █████ | 7.283%
/model.8/cv2/conv/Conv:                       | █████ | 7.092%
/model.10/m/m.0/attn/MatMul:                  | █████ | 6.654%
/model.8/cv1/conv/Conv:                       | █████ | 6.492%
/model.8/m.0/m/m.1/cv1/conv/Conv:             | █████ | 6.451%
/model.23/cv2.0/cv2.0.2/Conv:                 | ████ | 5.990%
/model.23/cv2.2/cv2.2.2/Conv:                 | ████ | 5.902%
/model.6/m.0/m/m.0/cv2/conv/Conv:             | ████ | 5.898%
/model.6/m.0/cv2/conv/Conv:                   | ████ | 5.881%
/model.6/m.0/cv3/conv/Conv:                   | ████ | 5.402%
/model.8/m.0/cv1/conv/Conv:                   | ████ | 5.210%
/model.23/cv3.2/cv3.2.2/Conv:                 | ████ | 5.126%
/model.6/cv1/conv/Conv:                       | ████ | 4.983%
/model.9/cv2/conv/Conv:                       | ███ | 4.616%
/model.9/cv1/conv/Conv:                       | ███ | 3.934%
/model.7/conv/Conv:                           | ███ | 3.906%
/model.3/conv/Conv:                           | ███ | 3.654%
```

(continues on next page)

(continued from previous page)

```
/model.6/cv2/conv/Conv:                          | ██                    | 3.429%
/model.8/m.0/m/m.0/cv1/conv/Conv:                | ██                    | 3.319%
/model.2/cv2/conv/Conv:                          | ██                    | 3.220%
/model.6/m.0/m/m.1/cv1/conv/Conv:                | ██                    | 3.191%
/model.6/m.0/m/m.0/cv1/conv/Conv:                | ██                    | 3.157%
/model.4/cv1/conv/Conv:                          | ██                    | 2.893%
/model.6/m.0/m/m.1/cv2/conv/Conv:                | ██                    | 2.792%
/model.6/m.0/cv1/conv/Conv:                      | ██                    | 2.761%
/model.5/conv/Conv:                              | ██                    | 2.629%
/model.4/cv2/conv/Conv:                          | ██                    | 2.298%
/model.2/cv1/conv/Conv:                          | ▉                     | 2.107%
/model.2/m.0/cv2/conv/Conv:                      | ▉                     | 2.095%
/model.4/m.0/cv1/conv/Conv:                      | ▉                     | 2.069%
/model.23/cv3.1/cv3.1.2/Conv:                    | ▉                     | 1.744%
/model.1/conv/Conv:                              | ▉                     | 1.631%
/model.2/m.0/cv1/conv/Conv:                      | ▉                     | 1.583%
/model.4/m.0/cv2/conv/Conv:                      | ▉                     | 1.126%
/model.23/cv3.0/cv3.0.2/Conv:                    |                       | 0.535%
/model.0/conv/Conv:                              |                       | 0.067%
Analysing Layerwise quantization error:: 100%|██████████| 98/98 [10:49<00:00,  6.63s/
↪it]
Layer                                            | NOISE:SIGNAL POWER RATIO
/model.9/cv2/conv/Conv:                          | ████████████████████  | 2.976%
/model.2/cv2/conv/Conv:                          | ██████████            | 1.610%
/model.3/conv/Conv:                              | ██████                | 0.854%
/model.2/cv1/conv/Conv:                          | ████                  | 0.543%
/model.1/conv/Conv:                              | ███                   | 0.487%
/model.8/cv1/conv/Conv:                          | ███                   | 0.414%
/model.4/cv2/conv/Conv:                          | ███                   | 0.397%
/model.0/conv/Conv:                              | ██                    | 0.364%
/model.6/m.0/cv3/conv/Conv:                      | ██                    | 0.230%
/model.5/conv/Conv:                              | ▉                     | 0.181%
/model.2/m.0/cv2/conv/Conv:                      | ▉                     | 0.144%
/model.13/cv2/conv/Conv:                         | ▉                     | 0.140%
/model.2/m.0/cv1/conv/Conv:                      | ▉                     | 0.138%
/model.4/cv1/conv/Conv:                          | ▉                     | 0.129%
/model.16/cv2/conv/Conv:                         | ▉                     | 0.122%
/model.23/cv4.2/cv4.2.0/conv/Conv:               | ▉                     | 0.120%
/model.4/m.0/cv1/conv/Conv:                      | ▉                     | 0.107%
/model.23/cv4.1/cv4.1.0/conv/Conv:               | ▉                     | 0.096%
/model.19/cv2/conv/Conv:                         | ▉                     | 0.078%
/model.23/cv2.2/cv2.2.2/Conv:                    | ▉                     | 0.076%
/model.4/m.0/cv2/conv/Conv:                      |                       | 0.071%
/model.8/m.0/m/m.1/cv1/conv/Conv:                |                       | 0.071%
/model.6/cv2/conv/Conv:                          |                       | 0.067%
/model.6/cv1/conv/Conv:                          |                       | 0.066%
/model.17/conv/Conv:                             |                       | 0.060%
/model.23/cv4.2/cv4.2.1/conv/Conv:               |                       | 0.057%
/model.22/m.0/m/m.1/cv1/conv/Conv:               |                       | 0.056%
/model.16/cv1/conv/Conv:                         |                       | 0.051%
/model.10/cv1/conv/Conv:                         |                       | 0.050%
/model.23/cv4.2/cv4.2.2/Conv:                    |                       | 0.046%
/model.22/cv2/conv/Conv:                         |                       | 0.044%
/model.7/conv/Conv:                              |                       | 0.043%
/model.10/m/m.0/attn/pe/conv/Conv:               |                       | 0.043%
/model.10/cv2/conv/Conv:                         |                       | 0.037%
/model.19/cv1/conv/Conv:                         |                       | 0.037%
```

(continues on next page)

(continued from previous page)

```
/model.8/cv2/conv/Conv:                          |                          | 0.036%
/model.13/cv1/conv/Conv:                         |                          | 0.036%
/model.6/m.0/m/m.1/cv1/conv/Conv:                |                          | 0.033%
/model.22/m.0/cv3/conv/Conv:                     |                          | 0.031%
/model.19/m.0/cv1/conv/Conv:                     |                          | 0.027%
/model.23/cv3.2/cv3.2.0/cv3.2.0.1/conv/Conv:     |                          | 0.026%
/model.8/m.0/cv1/conv/Conv:                      |                          | 0.025%
/model.19/m.0/cv2/conv/Conv:                     |                          | 0.025%
/model.8/m.0/cv3/conv/Conv:                      |                          | 0.024%
/model.10/m/m.0/attn/qkv/conv/Conv:              |                          | 0.023%
/model.8/m.0/m/m.0/cv1/conv/Conv:                |                          | 0.023%
/model.22/m.0/cv1/conv/Conv:                     |                          | 0.021%
/model.6/m.0/m/m.0/cv1/conv/Conv:                |                          | 0.021%
/model.23/cv2.0/cv2.0.0/conv/Conv:               |                          | 0.020%
/model.6/m.0/cv1/conv/Conv:                      |                          | 0.020%
/model.23/cv4.0/cv4.0.0/conv/Conv:               |                          | 0.019%
/model.9/cv1/conv/Conv:                          |                          | 0.018%
/model.23/cv4.1/cv4.1.2/Conv:                    |                          | 0.018%
/model.23/cv2.1/cv2.1.1/conv/Conv:               |                          | 0.018%
/model.13/m.0/cv1/conv/Conv:                     |                          | 0.016%
/model.23/cv2.1/cv2.1.0/conv/Conv:               |                          | 0.016%
/model.23/cv4.1/cv4.1.1/conv/Conv:               |                          | 0.016%
/model.16/m.0/cv2/conv/Conv:                     |                          | 0.015%
/model.10/m/m.0/attn/proj/conv/Conv:             |                          | 0.013%
/model.23/cv3.1/cv3.1.1/cv3.1.1.1/conv/Conv:     |                          | 0.013%
/model.8/m.0/m/m.0/cv2/conv/Conv:                |                          | 0.013%
/model.16/m.0/cv1/conv/Conv:                     |                          | 0.012%
/model.23/cv2.2/cv2.2.0/conv/Conv:               |                          | 0.011%
/model.20/conv/Conv:                             |                          | 0.011%
/model.22/m.0/m/m.0/cv1/conv/Conv:               |                          | 0.011%
/model.23/cv3.2/cv3.2.1/cv3.2.1.1/conv/Conv:     |                          | 0.011%
/model.8/m.0/m/m.1/cv2/conv/Conv:                |                          | 0.010%
/model.23/cv2.0/cv2.0.2/Conv:                    |                          | 0.009%
/model.10/m/m.0/attn/MatMul:                     |                          | 0.009%
/model.22/cv1/conv/Conv:                         |                          | 0.009%
/model.13/m.0/cv2/conv/Conv:                     |                          | 0.008%
/model.23/cv2.2/cv2.2.1/conv/Conv:               |                          | 0.008%
/model.23/cv2.1/cv2.1.2/Conv:                    |                          | 0.007%
/model.23/cv3.2/cv3.2.1/cv3.2.1.0/conv/Conv:     |                          | 0.007%
/model.22/m.0/m/m.1/cv2/conv/Conv:               |                          | 0.007%
/model.6/m.0/m/m.0/cv2/conv/Conv:                |                          | 0.006%
/model.22/m.0/m/m.0/cv2/conv/Conv:               |                          | 0.006%
/model.23/cv4.0/cv4.0.1/conv/Conv:               |                          | 0.005%
/model.23/cv3.2/cv3.2.0/cv3.2.0.0/conv/Conv:     |                          | 0.005%
/model.23/cv4.0/cv4.0.2/Conv:                    |                          | 0.004%
/model.6/m.0/m/m.1/cv2/conv/Conv:                |                          | 0.004%
/model.23/cv3.0/cv3.0.0/cv3.0.0.1/conv/Conv:     |                          | 0.004%
/model.10/m/m.0/ffn/ffn.1/conv/Conv:             |                          | 0.003%
/model.23/cv3.2/cv3.2.2/Conv:                    |                          | 0.003%
/model.10/m/m.0/attn/MatMul_1:                   |                          | 0.002%
/model.10/m/m.0/ffn/ffn.0/conv/Conv:             |                          | 0.002%
/model.23/cv3.1/cv3.1.0/cv3.1.0.1/conv/Conv:     |                          | 0.002%
/model.23/cv2.0/cv2.0.1/conv/Conv:               |                          | 0.002%
/model.23/cv3.1/cv3.1.1/cv3.1.1.0/conv/Conv:     |                          | 0.001%
/model.23/cv3.0/cv3.0.2/Conv:                    |                          | 0.001%
/model.23/cv3.1/cv3.1.2/Conv:                    |                          | 0.001%
/model.23/cv3.0/cv3.0.1/cv3.0.1.0/conv/Conv:     |                          | 0.001%
```

(continues on next page)

(continued from previous page)

```
/model.23/cv3.1/cv3.1.0/cv3.1.0.0/conv/Conv: |                    | 0.001%
/model.23/cv3.0/cv3.0.0/cv3.0.0.0/conv/Conv: |                    | 0.000%
/model.6/m.0/cv2/conv/Conv:                  |                    | 0.000%
/model.23/cv3.0/cv3.0.1/cv3.0.1.1/conv/Conv: |                    | 0.000%
/model.8/m.0/cv2/conv/Conv:                  |                    | 0.000%
/model.22/m.0/cv2/conv/Conv:                 |                    | 0.000%
```

Quantization error analysis

After applying QAT to 8-bit quantization, the quantized model's Pose mAP50:95 on COCO improves to 44.9% with the same inputs, while cumulative errors of out layers are significantly reduced. Compared to the other two quantization methods, the 8-bit QAT quantized model achieves the highest quantization accuracy with the lowest inference latency.

3.8.3 Model deployment

example

Object detection base class

- dl_detect_base.hpp
- dl_detect_base.cpp

Pre-process

`ImagePreprocessor` class contains the common pre-precoess pipeline, `color conversion`, `crop`, `resize`, `normalization`, `quantize`

- dl_image_preprocessor.hpp
- dl_image_preprocessor.cpp

Post-process

- dl_detect_postprocessor.hpp
- dl_detect_postprocessor.cpp
- dl_pose_yolo11_postprocessor.hpp
- dl_pose_yolo11_postprocessor.cpp

3.9 How to Deploy Streaming Models

Time series models are now widely applied in various fields, such as audio processing. Audio models typically have two deployment modes when deployed:

- Offline mode: The model receives the complete audio data (e.g., an entire speech file) at once and processes it as a whole.

- Streaming mode: In streaming mode, the model receives audio data frame by frame (or chunk by chunk) in real-time, processes it, and outputs intermediate results.

In this tutorial, we will introduce how to quantize a streaming model using ESP-PPQ and deploy the quantized streaming model with ESP-DL.

3.9.1 Prerequisites

1. *Install ESP-IDF*
2. *Install ESP-PPQ*

3.9.2 Model Quantization

Reference example

There are numerous types of time series models. Here, we take the Temporal Convolutional Network (TCN) as an example. If you are unfamiliar with TCNs, please refer to relevant resources for details; we won't elaborate further. Other models should be customized based on their specific structures.

The example code constructs a TCN model: models.py (the model is incomplete and used only for demonstration).

ESP-PPQ provides an automatic streaming conversion feature that simplifies the process of creating streaming models. With the `auto_streaming=True` parameter, ESP-PPQ automatically handles the model transformation required for streaming inference.

Note:

- In offline mode, the model input is a complete data segment, and the input shape typically has a large size along the time dimension (e.g., `[1, 16, 15]`).
- In streaming mode, the model input is continuous data with a smaller time dimension, which matches the chunk size for real-time processing (e.g., `[1, 16, 3]`).

Automatic Streaming Conversion

ESP-PPQ provides an automatic streaming conversion feature via the `auto_streaming=True` parameter in the quantization process. When this flag is enabled, ESP-PPQ automatically transforms the model to support streaming inference by:

1. Analyzing the model structure to identify appropriate chunking points
2. Creating internal state management for maintaining context between chunks
3. Generating optimized code suitable for streaming scenarios

How Auto Streaming Conversion Works

The automatic streaming conversion in ESP-PPQ analyzes the model graph and inserts `StreamingCache` nodes at strategic locations to enable temporal context preservation. The conversion process follows these principles:

1. Operation Classification

- **Streaming-enabled operations**: Convolution, pooling, and transpose convolution operations that require temporal context (e.g., `Conv`, `AveragePool`, `MaxPool`, `ConvTranspose`).
- **Bypass operations**: Activation functions, mathematical operations, quantization nodes, and other operations that don't require temporal context (e.g., `Relu`, `Add`, `MatMul`, `LayerNorm`).

2. Window Size Calculation

For streaming-enabled operations, ESP-PPQ calculates the required cache window size based on: - Kernel size and dilation rates - Padding configuration - Stride values

The window size determines how many previous frames need to be cached for proper computation of the current frame.

3. StreamingCache Node Insertion

ESP-PPQ inserts `StreamingCache` nodes before streaming-enabled operations. These nodes: - Maintain a sliding window buffer of historical frames - Adjust tensor shapes to accommodate the cache window - Preserve quantization configurations from the original operation - Manage frame axis alignment for proper temporal processing

4. Padding Adjustment

For streaming operations, ESP-PPQ adjusts padding configurations: - Removes bottom padding to prevent look-ahead into future frames - Maintains symmetric or top-only padding for causal processing

Limitations and Considerations

- Automatic conversion supports convolution-based temporal operations out-of-the-box
- Custom operations or complex temporal dependencies may require manual streaming table configuration
- The conversion assumes the time dimension is along axis 1 (configurable via `streaming_table`)

Here's an example of how to use the auto streaming feature:

```
# Export non-streaming model
quant_ppq_graph = espdl_quantize_torch(
    model=model,
    espdl_export_file=ESPDL_MODEL_PATH,
    calib_dataloader=dataloader,
    calib_steps=32,  # Number of calibration steps
    input_shape=INPUT_SHAPE,  # Input shape for offline mode
    inputs=None,
    target=TARGET,  # Quantization target type
```

(continues on next page)

(continued from previous page)

```
    num_of_bits=NUM_OF_BITS,  # Number of quantization bits
    dispatching_override=None,
    device=DEVICE,
    error_report=True,
    skip_export=False,
    export_test_values=True,
    verbose=1,  # Output detailed log information
)

# Export streaming model with automatic conversion
quant_ppq_graph = espdl_quantize_torch(
    model=model,
    espdl_export_file=ESPDL_STEAMING_MODEL_PATH,
    calib_dataloader=dataloader,
    calib_steps=32,
    input_shape=INPUT_SHAPE,
    inputs=None,
    target=TARGET,
    num_of_bits=NUM_OF_BITS,
    dispatching_override=None,
    device=DEVICE,
    error_report=True,
    skip_export=False,
    export_test_values=False,
    verbose=1,
    auto_streaming=True,  # Enable automatic streaming conversion
    streaming_input_shape=[1, 16, 3],  # Input shape for streaming mode
    streaming_table=None,
)
```

Manual Streaming Cache Configuration

For operators that are not automatically supported by ESP-PPQ's streaming conversion feature (such as Transpose, Reshape, Slice, etc.), you can manually insert StreamingCache nodes using the `insert_streaming_cache_on_var` function. This function allows you to specify cache attributes for variables that cannot have streamingCache inserted automatically.

The `insert_streaming_cache_on_var` function has the following signature:

```
def insert_streaming_cache_on_var(
    var_name: str,
    window_size: int,
    op_name: str = None,
    frame_axis: int = 1
) -> Dict[str, Any]
```

Parameters: - `var_name`: The name of the variable where the streaming cache should be inserted - `window_size`: The size of the cache window (number of frames to cache) - `op_name`: (Optional) The name of the operator associated with the variable - `frame_axis`: (Optional) The axis representing the time dimension, default is 1

The function returns a dictionary containing the streaming cache configuration, which should be added to a `streaming_table` list and passed to the `espdl_quantize_torch` function.

Example usage:

```
streaming_table = []
# Manually specify cache attributes for variables that cannot insert streamingCache␣
↪automatically
streaming_table.append(
    insert_streaming_cache_on_var("/out_conv/Conv_output_0", output_frame_size - 1)
)
streaming_table.append(insert_streaming_cache_on_var("PPQ_Variable_0", 1, "/Slice"))

quant_ppq_graph = espdl_quantize_torch(
    model=model,
    espdl_export_file=ESPDL_STEAMING_MODEL_PATH,
    calib_dataloader=dataloader,
    calib_steps=32,
    input_shape=INPUT_SHAPE,
    inputs=None,
    target=TARGET,
    num_of_bits=NUM_OF_BITS,
    dispatching_override=None,
    device=DEVICE,
    error_report=True,
    skip_export=False,
    export_test_values=False,
    verbose=1,
    auto_streaming=True,
    streaming_input_shape=[1, 16, 3],
    streaming_table=streaming_table,  # Pass the manually configured streaming table
)
```

3.9.3 Model Deployment

Reference example , this example uses pre-generated data to simulate a real-time data stream.

Note: For basic model loading and inference methods, please refer to other documents:

- *How to Load and Test a Model*
- *How to Perform Model Inference*

In streaming mode, the model receives data in chunks over time rather than requiring the entire input at once. The streaming model processes these chunks sequentially while maintaining internal state between chunks. The deployment code handles splitting the input into appropriate chunks and feeding them to the model. See app_main.cpp for the following code block:

```
dl::TensorBase *run_streaming_model(dl::Model *model, dl::TensorBase *test_input)
{
    std::map<std::string, dl::TensorBase *> model_inputs = model->get_inputs();
    dl::TensorBase *model_input = model_inputs.begin()->second;
    std::map<std::string, dl::TensorBase *> model_outputs = model->get_outputs();
    dl::TensorBase *model_output = model_outputs.begin()->second;

    if (!test_input) {
        ESP_LOGE(TAG,
                 "Model input doesn't have a corresponding test input. Please enable␣
↪export_test_values option "
                 "in esp-ppq when export espdl model.");
```

(continues on next page)

(continued from previous page)

```
        return nullptr;
    }

    int test_input_size = test_input->get_bytes();
    uint8_t *test_input_ptr = (uint8_t *)test_input->data;
    int model_input_size = model_input->get_bytes();
    uint8_t *model_input_ptr = (uint8_t *)model_input->data;
    int chunks = test_input_size / model_input_size;
    for (int i = 0; i < chunks; i++) {
        // assign chunk data to model input
        memcpy(model_input_ptr, test_input_ptr + i * model_input_size, model_input_
↪size);
        model->run(model_input);
    }

    return model_output;
}
```

This approach allows the model to process long sequences efficiently by breaking them into smaller, manageable chunks. Each chunk is fed to the model sequentially, and the internal state is maintained automatically to ensure continuity across chunks.

Note:

- The number of chunks is calculated based on the ratio between the full input size and the streaming model's input size.
- ESP-DL streaming models handle internal state management automatically, making deployment straightforward.
- The output from the streaming model should match the final portion of the equivalent offline model's output.

3.10 Quantizing Models with TQT

This document explains why and how to use Trained Quantization Thresholds (TQT) in ESP-PPQ for quantization to achieve higher quantized accuracy. Make sure your ESP-PPQ installation is updated to at least version 1.2.7.

3.10.1 Why Use TQT

TQT (Trained Quantization Thresholds) comes from the paper Trained Quantization Thresholds for Accurate and Efficient Fixed-Point Inference of Deep Neural Networks (Sambhav R. Jain, Albert Gural, Michael Wu, Chris H. Dick), published at **MLSys 2020**. Since ESP32-series chips currently only support `.espdl` models produced by **Per-Tensor + Symmetric + Power-of-2** quantization, we have integrated the core idea of TQT into ESP-PPQ, jointly optimizing quantization thresholds (scale) and model weights via **standard backpropagation and gradient descent** to match hardware constraints.

In ESP-PPQ, TQT is implemented as **TrainedQuantizationThresholdPass**. We optimize scale in the log domain and adapt to ESP-DL's **Power-of-2** constraint (e.g. int_lambda, STE, etc.).

Limitations of Post-Training Quantization (PTQ)

Method	Pros	Limitations
PTQ	Simple, no training	Scale is statistics-based and may be suboptimal; larger errors on sensitive structures (e.g. YOLO26n head)
TQT	Finetune scale/weights	Requires calibration data and compute

Complexity of Quantization-Aware Training (QAT)

Quantization-aware training (QAT) usually inserts fake quantization in full training or finetuning and can achieve good accuracy, but it requires:

- **Full labeled data and long training/finetuning** (even for models that do not use labels, QAT often needs long training/finetuning);
- **More compute and hyperparameter tuning**;
- Extra engineering to integrate with existing PTQ/calibration pipelines.

TQT, by contrast, **does not need labels**: the loss is MSE between floating-point output and quantized output; only a calibration set is needed. Under the Power-of-2 constraint it jointly finetunes weights and log2(scale), and often improves accuracy in relatively few steps.

What TQT Offers

- **Learnable scale (still 2^k)**: Optimize alpha = log2(scale) in the log domain; numerically stable and naturally satisfies POWER_OF_2. Training will be skipped if the conditions are not met or if disabled.
- **STE and range–precision tradeoff**: Using STE for threshold gradients and a sensible forward/backward design gives a better tradeoff between **representation range** and **accuracy**; in ESP-PPQ this is done via alpha_ste.
- **Learnable weights**: Weights are trainable and can be fine-tuned together with scales on a block-wise basis to further reduce quantization error.
- **Alignment with ESP-DL**: With the default alpha_ste forward, the forward uses integer exponents; with int_lambda the learned alpha is closer to an integer, which eases export and matches the chip side.
- **No labels**: Only a calibration set is needed for scale/weight finetuning.

3.10.2 How to Use TQT

Quick Start

1. Use ESP-DL default setting and enable TQT

```
from esp_ppq import QuantizationSettingFactory
from esp_ppq.api import espdl_quantize_onnx

quant_setting = QuantizationSettingFactory.espdl_setting()
quant_setting.tqt_optimization = True
quant_setting.tqt_optimization_setting.int_lambda = 0.25   # optional: make exponent␣
↪closer to integer
quant_setting.tqt_optimization_setting.steps = 500
quant_setting.tqt_optimization_setting.lr = 1e-5
quant_setting.tqt_optimization_setting.collecting_device = "cuda"  # or "cpu"
quant_setting.tqt_optimization_setting.block_size = 2
```

2. Pass the setting into quantization

```
from esp_ppq.api import ENABLE_CUDA_KERNEL
with ENABLE_CUDA_KERNEL():
    quant_ppq_graph = espdl_quantize_onnx(
        onnx_import_file=ONNX_PATH,
        espdl_export_file=ESPDL_MODLE_PATH,
        calib_dataloader=dataloader,
        calib_steps=32,
        input_shape=[1] + INPUT_SHAPE,
        target=TARGET,
        num_of_bits=NUM_OF_BITS,
        collate_fn=collate_fn,
        setting=quant_setting,
        device=DEVICE,
        error_report=False,
        skip_export=False,
        export_test_values=True,
        verbose=0,
        inputs=None,
    )
```

3. Export and return value

- **Returned quant_ppq_graph**: scale is **TQT-finetuned 2^integer**, and can be used for model accuracy evaluation.
- **Exported .espdl file**: exponent comes from int(log2(config.scale)); **export and chip-side inference use the TQT-optimized scale**.

TQTSetting Parameters

Parameter	Type	Default	Description
interested_layers	List[str]	[]	Names of ops to finetune; if empty, all eligible Conv/Gemm etc. are used
steps	int	500	Finetuning steps per block
lr	float	1e-5	Learning rate
block_size	int	4	Block size for graph splitting; affects stability and speed
is_scale_trainable	bool	True	Whether to finetune scale (POWER_OF_2 + LINEAR + SYMMETRICAL). True: train scale and weights; False: train only weights
gamma	float	0.0	Regularization: MSE(floating-point output, quantized output)
int_lambda	float	0.0	Regularization: pull alpha toward round(alpha); range [0.0, 1.0]
collecting_device	str	cpu	Device for caching calibration data; set to cuda if GPU is available

3.10.3 TQT Quantization Examples

The following examples show how to enable TQT quantization for **YOLO26n** (detection) and **MobileNetV2** (classification) and export to ESP-DL. Adjust calibration data and model paths to your environment.

YOLO26n Quantization

Preparation

Item	Description
Model name	YOLO26n (one2one branch end2end inference)
Task	Object detection
Input shape	[1, 3, 640, 640] (NCHW)
ONNX	yolo26n_o2o.onnx
Calibration	calib_yolo26n.zip
Note	Non-end2end with one2many is also supported; ONNX: yolo26n_o2m.onnx; quantization method is the same as end2end

Quantization script

```
import os
from esp_ppq import QuantizationSettingFactory
from esp_ppq.api import espdl_quantize_onnx
from torch.utils.data import DataLoader
import torch
from torch.utils.data import Dataset
from torchvision import transforms
from PIL import Image
from onnxsim import simplify
import onnx
import zipfile
import urllib.request
from esp_ppq.api import ENABLE_CUDA_KERNEL

class CaliDataset(Dataset):
    def __init__(self, path, img_shape=640):
        super().__init__()
        self.transform = transforms.Compose(
            [
                transforms.Resize((img_shape, img_shape)),
                transforms.ToTensor(),
                transforms.Normalize(mean=[0, 0, 0], std=[1, 1, 1]),
            ]
        )

        self.imgs_path = []
        self.path = path
        for img_name in os.listdir(self.path):
            img_path = os.path.join(self.path, img_name)
            self.imgs_path.append(img_path)

    def __len__(self):
        return len(self.imgs_path)

    def __getitem__(self, idx):
        img = Image.open(self.imgs_path[idx])
        if img.mode == 'L':
            img = img.convert('RGB')
        img = self.transform(img)
        return img


def report_hook(blocknum, blocksize, total):
    downloaded = blocknum * blocksize
    percent = downloaded / total * 100
    print(f"\rDownloading calibration dataset: {percent:.2f}%", end="")


def quant_yolo26n(imgsz):
    BATCH_SIZE = 32
    INPUT_SHAPE = [3, imgsz, imgsz]
    DEVICE = "cpu"
    TARGET = "esp32p4" # or "esp32s3"
    NUM_OF_BITS = 8
```

(continues on next page)

(continued from previous page)

```
    yolo26n_onnx_url = "https://dl.espressif.com/public/yolo26n_o2o.onnx"
    ONNX_PATH = "yolo26n_o2o.onnx"
    urllib.request.urlretrieve(
        yolo26n_onnx_url, "yolo26n_o2o.onnx", reporthook=report_hook
    )

    ESPDL_MODLE_PATH = "yolo26n_o2o_ptq_fq_tqt_p4_640.espdl"

    yolo26n_caib_url = "https://dl.espressif.com/public/calib_yolo26n.zip"
    CALIB_DIR = "calib_yolo26n"
    urllib.request.urlretrieve(
        yolo26n_caib_url, "calib_yolo26n.zip", reporthook=report_hook
    )
    with zipfile.ZipFile("calib_yolo26n.zip", "r") as zip_file:
        zip_file.extractall("./")

    model = onnx.load(ONNX_PATH)
    sim = True
    if sim:
        model, check = simplify(model)
        assert check, "Simplified ONNX model could not be validated"
    onnx.save(onnx.shape_inference.infer_shapes(model), ONNX_PATH)

    calibration_dataset = CaliDataset(CALIB_DIR, img_shape=imgsz)
    dataloader = DataLoader(
        dataset=calibration_dataset, batch_size=BATCH_SIZE, shuffle=False, num_
↪workers=8,
    )

    def collate_fn(batch: torch.Tensor) -> torch.Tensor:
        return batch.to(DEVICE)


    # default setting
    quant_setting = QuantizationSettingFactory.espdl_setting()
    # activation calibration algo
    quant_setting.quantize_activation_setting.calib_algorithm = "percentile"  # kl -->
↪ percentile to get better mAP
    # focused logits quantization
    quant_setting.quant_config_modify = True
    # o2o
    quant_setting.quant_config_modify_setting.custom_config = {
        "/model.23/one2one_cv3.0/one2one_cv3.0.2/Conv": 0.0625,
        "/model.23/one2one_cv3.1/one2one_cv3.1.2/Conv": 0.0625,
        "/model.23/one2one_cv3.2/one2one_cv3.2.2/Conv": 0.0625,
    }

    # o2m (alternative)
    # quant_setting.quant_config_modify_setting.custom_config = {
    #     "/model.23/cv3.0/cv3.0.2/Conv": 0.0625,
    #     "/model.23/cv3.1/cv3.1.2/Conv": 0.0625,
    #     "/model.23/cv3.2/cv3.2.2/Conv": 0.0625,
    # }

    # TQT
    quant_setting.tqt_optimization = True
    quant_setting.tqt_optimization_setting.collecting_device = "cpu"
```

(continues on next page)

(continued from previous page)

```
    quant_setting.tqt_optimization_setting.steps = 200  #300 for o2m
    quant_setting.tqt_optimization_setting.block_size = 2
    quant_setting.tqt_optimization_setting.lr = 1e-5

    quant_ppq_graph = espdl_quantize_onnx(
        onnx_import_file=ONNX_PATH,
        espdl_export_file=ESPDL_MODLE_PATH,
        calib_dataloader=dataloader,
        calib_steps=32,
        input_shape=[1] + INPUT_SHAPE,
        target=TARGET,
        num_of_bits=NUM_OF_BITS,
        collate_fn=collate_fn,
        setting=quant_setting,
        device=DEVICE,
        error_report=True,
        skip_export=False,
        export_test_values=False,
        verbose=0,
        inputs=None,
    )
    return quant_ppq_graph


if __name__ == "__main__":
    quant_yolo26n(imgsz=640)
```

Focused Logits Quantization

In detection models, unlike backbone/neck, **detection head outputs** are very sensitive to quantization error. For models like YOLO26n, the classification branch outputs logits that are **inputs to Sigmoid**; after Sigmoid they become class confidence or scores. Using the same calibration-derived scale for these logits as for other layers often leads to **scale that is too large (quantization step too coarse)**, distorting class scores and reducing mAP.

- **Sigmoid has a limited effective input range**: σ(x) = 1/(1+exp(-x)) saturates quickly for large absolute x—output tends to 0 when x is very negative and to 1 when very positive; only the middle range (roughly -4 to 4) changes meaningfully. So the logits that actually discriminate are in this **limited range**; outside it they sit in saturation and barely change.
- **Saturation on both sides; detection cares more about positive samples**: In object detection we care more about **positive samples** (with object, high confidence), i.e. logits on the positive side and Sigmoid output near 1. On both sides, if logits are quantized too coarsely (scale too large, step too big), many different logits map to the same quantized level: near saturation they get "squashed" to 0 or 1, and in the effective middle range fine-grained distinction is lost, so scores are distorted and mAP drops.

So for **logits fed into Sigmoid** (the last Conv outputs of the three head branches one2one_cv3.0/1/2.2 in the example), we use **Focused logits quantization**: assign **finer scale** (e.g. 0.0625 = 2^-4) to these layers so that, while keeping Power-of-2, the quantization step is smaller, more levels are kept in Sigmoid's effective range, saturation is delayed, and discrimination on the positive side is preserved, stabilizing or improving mAP. For other detection models or different export structures, identify which layers feed Sigmoid and adjust custom_config accordingly.

Quantization accuracy on ESP32-P4

Accuracy after quantizing yolo26n (size=640 pixels) on COCO val2017:

Config	mAP50-95(o2m)	mAP50-95(e2e)	Note
PTQ (no TQT)	0.342	0.332	Calibration only
PTQ + TQT	0.371	0.363	Calibration + TQT scale/weight finetuning

Results show that TQT significantly improves mAP for both end2end and non-end2end models.

Note: For faster inference, you can use size=512 pixels. Experiments show that with size=512 pixels and end2end inference, the 8-bit PTQ model reaches mAP50-95 of 0.315. After TQT, mAP50-95 improves to 0.341, representing a gain of 2.6 percentage points.

MobileNetV2 Quantization

Quantization script

```
import os
import subprocess
from typing import Iterable, Tuple, List, Tuple

import torch
from datasets.imagenet_util import (
    evaluate_ppq_module_with_imagenet,
    load_imagenet_from_directory,
)
from esp_ppq import QuantizationSettingFactory, QuantizationSetting
from esp_ppq.api import espdl_quantize_onnx, get_target_platform
from torch.utils.data import DataLoader
import torchvision.datasets as datasets
import torchvision.transforms as transforms
from torch.utils.data.dataset import Subset
import urllib.request
import zipfile


def quant_setting_mobilenet_v2(
    onnx_path: str,
    optim_quant_method: List[str] = None,
) -> Tuple[QuantizationSetting, str]:
    '''
    Quantize onnx model with optim_quant_method.

    Args:
        optim_quant_method (List[str]): support 'MixedPrecision_quantization',
↪'LayerwiseEqualization_quantization'
        -'MixedPrecision_quantization': if some layers in model have larger errors in␣
↪8-bit quantization, dispathching
                                        the layers to 16-bit quantization. You can␣
```

(continues on next page)

(continued from previous page)

```
↪remove or add layers according to your
                                        needs.
        -'LayerwiseEqualization_quantization' using weight equalization strategy,␣
↪which is proposed by Markus Nagel.
                                                    Refer to paper https://openaccess.
↪thecvf.com/content_ICCV_2019/papers/Nagel_Data-Free_Quantization_Through_Weight_
↪Equalization_and_Bias_Correction_ICCV_2019_paper.pdf for more information.
                                                    Since ReLU6 exists in MobilenetV2,␣
↪convert ReLU6 to ReLU for better precision.

    Returns:
        [tuple]: [QuantizationSetting, str]
    '''
    quant_setting = QuantizationSettingFactory.espdl_setting()
    if optim_quant_method is not None:
        if "MixedPrecision_quantization" in optim_quant_method:
            # These layers have larger errors in 8-bit quantization, dispatching to␣
↪16-bit quantization.
            # You can remove or add layers according to your needs.
            quant_setting.dispatching_table.append(
                "/features/features.1/conv/conv.0/conv.0.0/Conv",
                get_target_platform(TARGET, 16),
            )
            quant_setting.dispatching_table.append(
                "/features/features.1/conv/conv.0/conv.0.2/Clip",
                get_target_platform(TARGET, 16),
            )
        elif "LayerwiseEqualization_quantization" in optim_quant_method:
            # layerwise equalization
            quant_setting.equalization = True
            quant_setting.equalization_setting.iterations = 4
            quant_setting.equalization_setting.value_threshold = 0.4
            quant_setting.equalization_setting.opt_level = 2
            quant_setting.equalization_setting.interested_layers = None
            # replace ReLU6 with ReLU
            onnx_path = onnx_path.replace("mobilenet_v2.onnx", "mobilenet_v2_relu.onnx
↪")
        else:
            raise ValueError(
                "Please set optim_quant_method correctly. Support 'MixedPrecision_
↪quantization', 'LayerwiseEqualization_quantization'"
            )

    return quant_setting, onnx_path


def collate_fn1(x: Tuple) -> torch.Tensor:
    return torch.cat([sample[0].unsqueeze(0) for sample in x], dim=0)


def collate_fn2(batch: torch.Tensor) -> torch.Tensor:
    return batch.to(DEVICE)


def report_hook(blocknum, blocksize, total):
    downloaded = blocknum * blocksize
    percent = downloaded / total * 100
```

(continues on next page)

(continued from previous page)

```
    print(f"\rDownloading calibration dataset: {percent:.2f}%", end="")


if __name__ == "__main__":
    BATCH_SIZE = 32
    INPUT_SHAPE = [3, 224, 224]
    DEVICE = "cpu"  #  'cuda' or 'cpu', if you use cuda, please make sure that cuda␣
↪is available
    TARGET = "esp32p4"  #  'c', 'esp32s3' or 'esp32p4'
    NUM_OF_BITS = 8
    ONNX_PATH = "./models/torch/mobilenet_v2.onnx"  #'models/onnx/mobilenet_v2.onnx'
    ESPDL_MODEL_PATH = "models/onnx/mobilenet_v2.espdl"
    CALIB_DIR = "./imagenet"

    # Download mobilenet_v2 model from onnx models and dataset
    imagenet_url = "https://dl.espressif.com/public/imagenet_calib.zip"
    os.makedirs(CALIB_DIR, exist_ok=True)
    if not os.path.exists("imagenet_calib.zip"):
        urllib.request.urlretrieve(
            imagenet_url, "imagenet_calib.zip", reporthook=report_hook
        )
    if not os.path.exists(os.path.join(CALIB_DIR, "calib")):
        with zipfile.ZipFile("imagenet_calib.zip", "r") as zip_file:
            zip_file.extractall(CALIB_DIR)
    CALIB_DIR = os.path.join(CALIB_DIR, "calib")

    # ------------------------------------------
    # Prepare Calibration Dataset
    # ------------------------------------------
    if os.path.exists(CALIB_DIR):
        print(f"load imagenet calibration dataset from directory: {CALIB_DIR}")
        dataset = datasets.ImageFolder(
            CALIB_DIR,
            transforms.Compose(
                [
                    transforms.Resize(256),
                    transforms.CenterCrop(224),
                    transforms.ToTensor(),
                    transforms.Normalize(
                        mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]
                    ),
                ]
            ),
        )
        dataset = Subset(dataset, indices=[_ for _ in range(0, 1024)])
        dataloader = DataLoader(
            dataset=dataset,
            batch_size=BATCH_SIZE,
            shuffle=False,
            num_workers=4,
            pin_memory=False,
            collate_fn=collate_fn1,
        )
    else:
        # Random calibration dataset only for debug
        print("load random calibration dataset")
```

(continues on next page)

(continued from previous page)

```
        def load_random_calibration_dataset() -> Iterable:
            return [torch.rand(size=INPUT_SHAPE) for _ in range(BATCH_SIZE)]

        # Load training data for creating a calibration dataloader.
        dataloader = DataLoader(
            dataset=load_random_calibration_dataset(),
            batch_size=BATCH_SIZE,
            shuffle=False,
        )

    # ------------------------------------------
    # Quantize ONNX Model.
    # ------------------------------------------

    # create a setting for quantizing your network with ESPDL.
    # if you don't need to optimize quantization, set the input 1 of the quant_
↪setting_mobilenet_v2 function None
    # Example: Using LayerwiseEqualization_quantization
    quant_setting, ONNX_PATH = quant_setting_mobilenet_v2(
        ONNX_PATH, None
    )

    # TQT
    quant_setting.tqt_optimization = True
    quant_setting.tqt_optimization_setting.collecting_device = "cpu"
    quant_setting.tqt_optimization_setting.steps = 500
    quant_setting.tqt_optimization_setting.block_size = 4
    quant_setting.tqt_optimization_setting.lr = 1e-4

    quant_ppq_graph = espdl_quantize_onnx(
        onnx_import_file=ONNX_PATH,
        espdl_export_file=ESPDL_MODEL_PATH,
        calib_dataloader=dataloader,
        calib_steps=32,
        input_shape=[1] + INPUT_SHAPE,
        target=TARGET,
        num_of_bits=NUM_OF_BITS,
        collate_fn=collate_fn2,
        setting=quant_setting,
        device=DEVICE,
        error_report=True,
        skip_export=False,
        export_test_values=False,
        verbose=1,
    )

    # ------------------------------------------
    # Evaluate Quantized Model.
    # ------------------------------------------

    evaluate_ppq_module_with_imagenet(
        model=quant_ppq_graph,
        imagenet_validation_dir=CALIB_DIR,
        batchsize=BATCH_SIZE,
        device=DEVICE,
        verbose=1,
    )
```

Quantization results and analysis

When using only the TQT strategy for 8-bit quantization, the model achieves a Top-1 accuracy of 71.525%, representing a 1.725 percentage point improvement over the weight-equalization-only quantized model, which attains 69.800%. The result is also much closer to the floating-point baseline accuracy of 71.878%. These results demonstrate that even under strict 8-bit quantization constraints, TQT effectively mitigates performance degradation caused by quantization error by learning quantization thresholds, enabling the quantized model to approach float32-level accuracy.

3.10.4 FAQ

How to speed up TQT?

With a GPU, put calibration data on the GPU.

First set collecting_device to cuda:

```
quant_setting.tqt_optimization_setting.collecting_device = "cuda"
```

Then run quantization inside ENABLE_CUDA_KERNEL:

```
with ENABLE_CUDA_KERNEL():
        quant_ppq_graph = espdl_quantize_onnx(
            onnx_import_file=ONNX_PATH,
            espdl_export_file=ESPDL_MODLE_PATH,
            calib_dataloader=dataloader,
            calib_steps=32,
            input_shape=[1] + INPUT_SHAPE,
            target=TARGET,
            num_of_bits=NUM_OF_BITS,
            collate_fn=collate_fn,
            setting=quant_setting,
            device=DEVICE,
            error_report=False,
            skip_export=False,
            export_test_values=False,
            verbose=0,
            inputs=None,
        )
```

Increasing block_size (e.g. 2–4) reduces the number of blocks and total time, but too large can be unstable; try 4 first.

Can I train only model weights without modifying scale?

Yes. Set is_scale_trainable to False.

```
quant_setting.tqt_optimization_setting.is_scale_trainable = False
```

Can TQT be used together with Weight Equalization?

Yes. TQT can be applied in combination with Weight Equalization. For models such as espdet_pico, this combination can further enhance quantization performance.

```
quant_setting.tqt_optimization_setting.equalization = True
quant_setting.tqt_optimization_setting.tqt_optimization = True
```

CHAPTER

FOUR

API REFERENCE

4.1 Tensor API Reference

Tensor is the fundamental data type in esp-dl, used for storing multi-type data such as int8, int16, float, etc., similar to the tensor in PyTorch. We have implemented some common tensor operations. Please refer to the following APIs for details.

4.1.1 Header File

- esp-dl/dl/tensor/include/dl_tensor_base.hpp

4.1.2 Classes

class **ExponentInfo**

Exponent info for per-tensor / per-channel quantization.

Per-tensor: m_exponent only, zero heap allocation. Per-channel: heap-allocated m_exponents array. Provides operator int() so existing `DL_SCALE(tensor->exponent)` code works unchanged.

Public Functions

inline **ExponentInfo**(int exponent = 0)

Construct a per-tensor *ExponentInfo* with a single exponent value.

Parameters

exponent – The exponent value for per-tensor quantization. Default is 0.

inline **ExponentInfo**(const std::vector<int> &exponents)

Construct an *ExponentInfo* from a vector of exponents.

Parameters

exponents – Vector of exponent values. If size <= 1, uses per-tensor mode; otherwise uses per-channel mode with heap allocation.

inline **~ExponentInfo**()

Destroy the *ExponentInfo* object and free heap-allocated memory.

inline **ExponentInfo**(const *ExponentInfo* &other)

Copy constructor.

Parameters
other – The *ExponentInfo* object to copy from.

inline *ExponentInfo* &**operator=**(const *ExponentInfo* &other)

Copy assignment operator.

Parameters
other – The *ExponentInfo* object to assign from.

Returns
Reference to this object.

inline **ExponentInfo**(*ExponentInfo* &&other) noexcept

Move constructor.

Parameters
other – The *ExponentInfo* object to move from.

inline *ExponentInfo* &**operator=**(*ExponentInfo* &&other) noexcept

Move assignment operator.

Parameters
other – The *ExponentInfo* object to move from.

Returns
Reference to this object.

inline *ExponentInfo* &**operator=**(int value)

Assign a single integer value as per-tensor exponent.

Parameters
value – The exponent value to assign.

Returns
Reference to this object.

inline int **get**(int ch = -1) const

Get exponent value.

Parameters
ch – Channel index. -1 (default) returns per-tensor value. ch >= 0 returns per-channel value if available, otherwise per-tensor.

Returns
The exponent value for the specified channel or per-tensor exponent.

inline **operator int**() const

Implicit conversion to int, returns the per-tensor exponent value.

Returns
The per-tensor exponent value.

inline bool **is_per_channel**() const

Check if using per-channel quantization.

Returns
true if per-channel exponents are stored, false otherwise.

inline int **channel_size**() const

Get the number of channels.

Returns
Number of channels for per-channel mode, or 1 for per-tensor mode.

inline const int ***data**() const

Get pointer to exponent data.

Returns
Pointer to per-channel exponents array if available, otherwise pointer to per-tensor exponent.

inline bool **operator==**(const *ExponentInfo* &other) const

Compare two *ExponentInfo* objects for equality.

Parameters
other – The *ExponentInfo* object to compare with.

Returns
true if both have the same exponent values, false otherwise.

inline bool **operator!=**(const *ExponentInfo* &other) const

Compare two *ExponentInfo* objects for inequality.

Parameters
other – The *ExponentInfo* object to compare with.

Returns
true if exponent values differ, false if equal.

class **TensorBase**

This class is designed according to PyTorch Tensor. *TensorBase* is required to ensure that the first address are aligned to 16 bytes and the memory size should be a multiple of 16 bytes.

TODO:: Implement more functions

Public Functions

TensorBase(std::vector<int> shape, const void *element, int exponent = 0, dtype_t dtype = DATA_TYPE_FLOAT, bool deep = true, uint32_t caps = MALLOC_CAP_DEFAULT)

Construct a *TensorBase* object.

Parameters

- **shape** – Shape of tensor
- **element** – Pointer of data
- **exponent** – Exponent of tensor, default is 0
- **dtype** – Data type of element, default is float
- **deep** – True: malloc memory and copy data, false: use the pointer directly
- **caps** – Bitwise OR of MALLOC_CAP_* flags indicating the type of memory to be returned

TensorBase(std::vector<int> shape, const void *element, const std::vector<int> &exponents, dtype_t dtype = DATA_TYPE_FLOAT, bool deep = true, uint32_t caps = MALLOC_CAP_DEFAULT)

Construct a *TensorBase* object with per-channel exponents.

Parameters

- **shape** – Shape of tensor
- **element** – Pointer of data
- **exponents** – Per-channel exponents
- **dtype** – Data type of element, default is float
- **deep** – True: malloc memory and copy data, false: use the pointer directly
- **caps** – Bitwise OR of MALLOC_CAP_* flags indicating the type of memory to be returned

inline virtual **~TensorBase**()

Destroy the *TensorBase* object.

bool **assign**(*TensorBase* *tensor)

Assign tensor to this tensor.

Parameters
tensor –

Returns
true if assign successfully, otherwise false.

bool **assign**(std::vector<int> shape, const void *element, int exponent, dtype_t dtype)

Assign data to this tensor.

Parameters

- **shape** –
- **element** –
- **exponent** –
- **dtype** –

Returns
true if assign successfully, otherwise false.

inline int **get_size**()

Get the size of Tensor.

Returns
the size of Tensor.

inline int **get_aligned_size**()

Get the aligned size of Tensor.

Returns
the aligned size of Tensor.

inline size_t **get_dtype_bytes**()

Get the dtype size, in bytes.

Returns
the size of dtype.

inline const char ***get_dtype_string**()

Get the dtype string of Tensor.

Returns
the string of Tensor's dtype.

inline int **get_bytes**()

Get the bytes of Tensor.

Returns
the bytes of Tensor.

inline int **get_aligned_bytes**()

Get the bytes of Tensor.

Returns
the bytes of Tensor.

inline virtual void ***get_element_ptr**()

Get data pointer. If *cache(preload data pointer)* is not null, return cache pointer, otherwise return data pointer.

Returns
the pointer of Tensor's data

template<typename **T**>
inline *T* ***get_element_ptr**()

Get data pointer by the specified template. If *cache(preload data pointer)* is not null, return cache pointer, otherwise return data pointer.

Returns
the pointer of Tensor's data

TensorBase &**set_element_ptr**(void *data)

Set the data pointer of Tensor.

Parameters
data – point to data memory

Returns
TensorBase& self

inline std::vector<int> **get_shape**()

Get the shape of Tensor.

Returns
std::vector<int> the shape of Tensor

TensorBase &**set_shape**(const std::vector<int> shape)

Set the shape of Tensor.

Parameters
shape – the shape of Tensor.

Returns
Tensor.

inline int **get_exponent**()

Get the exponent of Tensor.

Returns
int the exponent of Tensor

inline dtype_t **get_dtype**()

Get the data type of Tensor.

Returns
dtype_t the data type of Tensor

inline uint32_t **get_caps**()

Get the memory flags of Tensor.

Returns
uint32_t the memory flags of Tensor

TensorBase ***reshape**(std::vector<int> shape)

Change a new shape to the Tensor without changing its data.

Parameters
shape – the target shape

Returns
TensorBase *self

template<typename **T**>
TensorBase ***flip**(const std::vector<int> &axes)

Flip the input Tensor along the specified axes.

Parameters
axes – the specified axes

Returns
TensorBase& self

TensorBase ***transpose**(*TensorBase* *input, std::vector<int> perm = {})

Reverse or permute the axes of the input Tensor.

Parameters

- **input** – the input Tensor
- **perm** – the new arrangement of the dims. if perm == {}, the dims arrangement will be reversed.

Returns
TensorBase *self

template<typename **T**>
TensorBase ***transpose**(*T* *input_element, std::vector<int> &input_shape, std::vector<int> &input_axis_offset, std::vector<int> &perm)

Reverse or permute the axes of the input Tensor.

Parameters

- **input_element** – the input data pointer
- **input_shape** – the input data shape
- **input_axis_offset** – the input data axis offset
- **perm** – the new arrangement of the dims. if perm == {}, the dims arrangement will be reversed.

Returns
TensorBase *self

bool **is_same_shape**(*TensorBase* *tensor)

Check the shape is the same as the shape of input.

Parameters
tensor – Input tensor pointer

Returns

- true: same shape
- false: not

bool **equal**(*TensorBase* *tensor, float epsilon = 1e-6, bool verbose = false)

Compare the shape and data of two Tensor.

Parameters

- **tensor** – Input tensor
- **epsilon** – The max error of two element
- **verbose** – If true, print the detail of results

Returns

true if two tensor is equal otherwise false

TensorBase ***slice**(const std::vector<int> &start, const std::vector<int> &end, const std::vector<int> &axes = {}, const std::vector<int> &step = {})

Produces a slice of the this tensor along multiple axes.

> **Warning:** The length of start, end and step must be same as the shape of input tensor

Parameters

- **start** – Starting indicesd
- **end** – Ending indices
- **axes** – Axes that starts and ends apply to.
- **step** – Slice step, step = 1 if step is not specified

Returns

TensorBase* Output tensor pointer, created by this slice function

template<typename **T**>
TensorBase ***pad**(*T* *input_element, const std::vector<int> &input_shape, const std::vector<int> &pads, const padding_mode_t mode, *TensorBase* *const_value = nullptr)

Pad input tensor.

Parameters

- **input_element** – Data pointer of input tensor
- **input_shape** – Shape of input tensor
- **pads** – The number of padding elements to add, pads format should be: [x1_begin, x2_begin, …, x1_end, x2_end,…]
- **mode** – Supported modes: constant(default), reflect, edge
- **const_value** – (Optional) A scalar value to be used if the mode chosen is constant

Returns

Output tensor pointer

TensorBase ***pad**(*TensorBase* *input, const std::vector<int> &pads, const padding_mode_t mode, *TensorBase* *const_value = nullptr)

Pad input tensor.

Parameters

- **input** – Input tensor pointer
- **pads** – Padding elements to add, pads format should be: [x1_begin, x2_begin, …, x1_end, x2_end,…]
- **mode** – Supported modes: constant(default), reflect, edge
- **const_value** – (Optional) A scalar value to be used if the mode chosen is constant

Returns
Output tensor pointer

template<typename **T**>
bool **compare_elements**(const *T* *gt_elements, float epsilon = 1e-6, bool verbose = false)

Compare the elements of two Tensor.

Parameters

- **gt_elements** – The ground truth elements
- **epsilon** – The max error of two element
- **verbose** – If true, print the detail of results

Returns
true if all elements are equal otherwise false

int **get_element_index**(const std::vector<int> &axis_index)

Get the index of element.

Parameters
axis_index – The coordinates of element

Returns
int the index of element

std::vector<int> **get_element_coordinates**(int index)

Get the coordinates of element.

Parameters
index – The index of element

Returns
The coordinates of element

template<typename **T**>
T **get_element**(int index)

Get a element of Tensor by index.

Parameters
index – The index of element

Returns
The element of tensor

template<typename **T**>

T **get_element**(const std::vector<int> &axis_index)

Get a element of Tensor.

Parameters
axis_index – The index of element

Returns
The element of tensor

void **memset**(int value)

Set a element of Tensor by index.

Parameters
value – The value of element

void **rand**()

Fill tensor data with random bytes from hardware RNG.

Uses esp_fill_random() to fill the tensor's internal buffer with random bytes. Requires ESP-IDF (esp_random.h).

size_t **set_preload_addr**(void *addr, size_t size)

Set preload address of Tensor.

Parameters

- **addr** – The address of preload data
- **size** – Size of preload data

Returns
The size of preload data

inline virtual void **preload**()

Preload the data of Tensor.

void **reset_bias_layout**(quant_type_t op_quant_type, bool is_depthwise)

Reset the layout of Tensor.

> **Warning:** Only available for Convolution. Don't use it unless you know exactly what it does.

Parameters

- **op_quant_type** – The quant type of operation
- **is_depthwise** – Whether is depthwise convolution

void **push**(*TensorBase* *new_tensor, int dim)

Push new_tensor to current tensor. The time series dimension size of new tensor must is lesser or equal than that of the current tensor.".

Parameters

- **new_tensor** – The new tensor will be pushed
- **dim** – Specify the dimension on which to perform streaming stack pushes

virtual void **print**(bool print_data = false)

print the information of *TensorBase*

Parameters

print_data – Whether print the data

Public Members

int **size**

size of element including padding

std::vector<int> **shape**

shape of Tensor

dtype_t **dtype**

data type of element

ExponentInfo **exponent**

exponent of element (per-tensor or per-channel)

bool **auto_free**

free element when object destroy

std::vector<int> **axis_offset**

element offset of each axis

void ***data**

data pointer

void ***cache**

cache pointer? used for preload and do not need to free

uint32_t **caps**

flags indicating the type of memory

Public Static Functions

static void **slice**(*TensorBase* *input, *TensorBase* *output, const std::vector<int> &start, const std::vector<int> &end, const std::vector<int> &axes = {}, const std::vector<int> &step = {})

Produces a slice along multiple axes.

Warning: The length of start, end and step must be same as the shape of input tensor

Parameters

- **input** – Input Tensor

- **output** – Output Tensor
- **start** – Starting indicesd
- **end** – Ending indices
- **axes** – Axes that starts and ends apply to.
- **step** – Slice step, step = 1 if step is not specified

4.2 Module API Reference

The Module is the base class for operators in esp-dl, and all operators inherit from this base class. This base class defines the basic interfaces for operators, enabling the model layer to automatically execute operators and manage memory planning.

4.2.1 Header File

- esp-dl/dl/module/include/dl_module_base.hpp

4.2.2 Classes

class **Module**

Base class for module.

Public Functions

Module(const char *name = NULL, module_inplace_t inplace = MODULE_NON_INPLACE, quant_type_t quant_type = QUANT_TYPE_NONE)

Construct a new *Module* object.

Parameters

- **name** – Name of module.
- **inplace** – Inplace operation mode
- **quant_type** – Quantization type

virtual **~Module**()

Destroy the *Module* object. Return resource.

inline virtual std::vector<int> **get_outputs_index**()

Get the tensor index of this module's outputs.

Returns

Tensor index of model's tensors

virtual std::vector<std::vector<int>> **get_output_shape**(std::vector<std::vector<int>> &input_shapes) = 0

Calculate output shape by input shape.

Parameters

input_shapes – Input shapes

Returns
outputs shapes

virtual void **forward**(*ModelContext* *context, runtime_mode_t mode = RUNTIME_MODE_AUTO) = 0

Build the module, high-level interface for *Module* layer.

Parameters

- **context** – *Model* context including all inputs and outputs and other runtime information
- **mode** – Runtime mode, default is RUNTIME_MODE_AUTO

inline virtual void **forward_args**(void *args)

Run the module, Low-level interface for base layer and multi-core processing.

Parameters
args – ArgsType, arithArgsType, resizeArgsType and so on

inline virtual void **print**()

print module information

inline virtual void **set_preload_addr**(void *addr, size_t size)

set preload RAM pointer

Parameters

- **addr** – Internal RAM address, should be aligned to 16 bytes
- **size** – The size of RAM address

inline virtual void **preload**()

Perform a preload operation.

> **Warning:** Not implemented

inline virtual void **reset**()

reset all state of module, include inputs? outputs and preload cache setting

virtual void **run**(*TensorBase* *input, *TensorBase* *output, runtime_mode_t mode = RUNTIME_MODE_SINGLE_CORE)

Run the module with single input and single output.

Parameters

- **input** – Input tensor
- **output** – Output tensor
- **mode** – Runtime mode

virtual void **run**(std::vector<dl::*TensorBase**> inputs, std::vector<dl::*TensorBase**> outputs, runtime_mode_t mode = RUNTIME_MODE_SINGLE_CORE)

Run the module by inputs and outputs.

Parameters

- **inputs** – Input tensors
- **outputs** – Output tensors
- **mode** – Runtime mode

Public Members

char ***name**

Name of module.

module_inplace_t **inplace**

Inplace type.

quant_type_t **quant_type**

Quantization type.

std::vector<int> **m_inputs_index**

Tensor index of model's tensors that used for inputs.

std::vector<int> **m_outputs_index**

Tensor index of model's tensors that used for outputs.

Public Static Functions

static inline *Module* ***deserialize**(fbs::*FbsModel* *fbs_model, std::string node_name)

create module instance by node serialization information

Parameters

- **fbs_model** – Flatbuffer's model
- **node_name** – The node name in model's graph

Returns

The pointer of module instance

4.2.3 Header File

- esp-dl/dl/module/include/dl_module_creator.hpp

4.2.4 Classes

class **ModuleCreator**

Singleton class for registering modules.

Public Types

using **Creator** = std::function<*Module**(fbs::*FbsModel**, std::string)>

Module creator function type.

Public Functions

inline void **register_module**(const std::string &op_type, *Creator* creator)

Register a module creator to the module creator map This function allows for the dynamic registration of new module types and their corresponding creator functions at runtime. By associating the module type name with the creator function, the system can flexibly create instances of various modules.

Parameters

- **op_type** – The module type name, used as the key in the map
- **creator** – The module creator function, used to create modules of a specific type

inline *Module* ***create**(fbs::*FbsModel* *fbs_model, const std::string &op_type, const std::string name)

Create module instance pointer.

Parameters

- **fbs_model** – Flatbuffer model pointer
- **op_type** – Module/Operator type
- **name** – *Module* name

Returns

Module instance pointer

inline void **register_dl_modules**()

Pre-register the already implemented modules.

inline void **print**()

Print all modules has been registered.

inline void **clear**()

Clear all modules has been registered.

Public Static Functions

static inline *ModuleCreator* ***get_instance**()

Get instance of *ModuleCreator* by this function. It is only safe method to get instance of *ModuleCreator* because *ModuleCreator* is a singleton class.

Returns

ModuleCreator instance pointer

4.3 Model API Reference

This section covers model loading and static memory planning, making it convenient for users to directly load and run ESPDL models.

4.3.1 Header File

- esp-dl/dl/model/include/dl_model_base.hpp

4.3.2 Macros

DL_LOG_INFER_LATENCY_INIT_WITH_SIZE(size)

DL_LOG_INFER_LATENCY_INIT()

DL_LOG_INFER_LATENCY_START()

DL_LOG_INFER_LATENCY_END()

DL_LOG_INFER_LATENCY_PRINT(prefix, key)

DL_LOG_INFER_LATENCY_END_PRINT(prefix, key)

DL_LOG_INFER_LATENCY_ARRAY_INIT_WITH_SIZE(n, size)

DL_LOG_INFER_LATENCY_ARRAY_INIT(n)

DL_LOG_INFER_LATENCY_ARRAY_START(i)

DL_LOG_INFER_LATENCY_ARRAY_END(i)

DL_LOG_INFER_LATENCY_ARRAY_PRINT(i, prefix, key)

DL_LOG_INFER_LATENCY_ARRAY_END_PRINT(i, prefix, key)

4.3.3 Classes

class **Model**

Neural Network *Model*.

Public Functions

Model(const char *rodata_address_or_partition_label_or_path, fbs::model_location_type_t location = fbs::MODEL_LOCATION_IN_FLASH_RODATA, int max_internal_size = 0, memory_manager_t mm_type = MEMORY_MANAGER_GREEDY, const uint8_t *key = nullptr, bool param_copy = true)

Create the *Model* object by rodata address or partition label.

Parameters

- **rodata_address_or_partition_label_or_path** – The address of model data while location is MODEL_LOCATION_IN_FLASH_RODATA. The label of partition while location is MODEL_LOCATION_IN_FLASH_PARTITION. The path of model while location is MODEL_LOCATION_IN_SDCARD.
- **location** – The model location.
- **max_internal_size** – In bytes. Limit the max internal size usage. Only take effect when there's a PSRAM, and you want to alloc memory on internal RAM first.
- **mm_type** – Type of memory manager
- **key** – The key of encrypted model.
- **param_copy** – Set to false to avoid copy model parameters from FLASH to PSRAM. Only set this param to false when your PSRAM resource is very tight. This saves PSRAM and sacrifices the performance of model inference because the frequency of PSRAM is higher than FLASH. Only takes effect when MODEL_LOCATION_IN_FLASH_RODATA(CONFIG_SPIRAM_RODATA not set) or MODEL_LOCATION_IN_FLASH_PARTITION.

Model(const char *rodata_address_or_partition_label_or_path, int model_index, fbs::model_location_type_t location = fbs::MODEL_LOCATION_IN_FLASH_RODATA, int max_internal_size = 0, memory_manager_t mm_type = MEMORY_MANAGER_GREEDY, const uint8_t *key = nullptr, bool param_copy = true)

Create the *Model* object by rodata address or partition label.

Parameters

- **rodata_address_or_partition_label_or_path** – The address of model data while location is MODEL_LOCATION_IN_FLASH_RODATA. The label of partition while location is MODEL_LOCATION_IN_FLASH_PARTITION. The path of model while location is MODEL_LOCATION_IN_SDCARD.
- **model_index** – The model index of packed models.
- **location** – The model location.
- **max_internal_size** – In bytes. Limit the max internal size usage. Only take effect when there's a PSRAM, and you want to alloc memory on internal RAM first.
- **mm_type** – Type of memory manager
- **key** – The key of encrypted model.
- **param_copy** – Set to false to avoid copy model parameters from FLASH to PSRAM. Only set this param to false when your PSRAM resource is very tight. This saves PSRAM and sacrifices the performance of model inference because the frequency of PSRAM is higher than FLASH. Only takes effect when MODEL_LOCATION_IN_FLASH_RODATA(CONFIG_SPIRAM_RODATA not set) or MODEL_LOCATION_IN_FLASH_PARTITION.

Model(const char *rodata_address_or_partition_label_or_path, const char *model_name, fbs::model_location_type_t location = fbs::MODEL_LOCATION_IN_FLASH_RODATA, int max_internal_size = 0, memory_manager_t mm_type = MEMORY_MANAGER_GREEDY, const uint8_t *key = nullptr, bool param_copy = true)

Create the *Model* object by rodata address or partition label.

Parameters

- **rodata_address_or_partition_label_or_path** – The address of model data while location is MODEL_LOCATION_IN_FLASH_RODATA. The label of partition

while location is MODEL_LOCATION_IN_FLASH_PARTITION. The path of model while location is MODEL_LOCATION_IN_SDCARD.

- **model_name** – The model name of packed models.
- **location** – The model location.
- **max_internal_size** – In bytes. Limit the max internal size usage. Only take effect when there's a PSRAM, and you want to alloc memory on internal RAM first.
- **mm_type** – Type of memory manager
- **key** – The key of encrypted model.
- **param_copy** – Set to false to avoid copy model parameters from FLASH to PSRAM. Only set this param to false when your PSRAM resource is very tight. This saves PSRAM and sacrifices the performance of model inference because the frequency of PSRAM is higher than FLASH. Only takes effect when MODEL_LOCATION_IN_FLASH_RODATA(CONFIG_SPIRAM_RODATA not set) or MODEL_LOCATION_IN_FLASH_PARTITION.

Model(fbs::*FbsModel* *fbs_model, int internal_size = 0, memory_manager_t mm_type = MEMORY_MANAGER_GREEDY)

Create the *Model* object by fbs_model.

Parameters

- **fbs_model** – The fbs model.
- **internal_size** – Internal ram size, in bytes
- **mm_type** – Type of memory manager

virtual **~Model**()

Destroy the *Model* object.

virtual esp_err_t **load**(const char *rodata_address_or_partition_label_or_path, fbs::model_location_type_t location = fbs::MODEL_LOCATION_IN_FLASH_RODATA, const uint8_t *key = nullptr, bool param_copy = true)

Load model graph and parameters from FLASH or sdcard.

Parameters

- **rodata_address_or_partition_label_or_path** – The address of model data while location is MODEL_LOCATION_IN_FLASH_RODATA. The label of partition while location is MODEL_LOCATION_IN_FLASH_PARTITION. The path of model while location is MODEL_LOCATION_IN_SDCARD.
- **location** – The model location.
- **key** – The key of encrypted model.
- **param_copy** – Set to false to avoid copy model parameters from FLASH to PSRAM. Only set this param to false when your PSRAM resource is very tight. This saves PSRAM and sacrifices the performance of model inference because the frequency of PSRAM is higher than FLASH. Only takes effect when MODEL_LOCATION_IN_FLASH_RODATA(CONFIG_SPIRAM_RODATA not set) or MODEL_LOCATION_IN_FLASH_PARTITION.

Returns

- ESP_OK Success
- ESP_FAIL Failed

virtual esp_err_t **load**(const char *rodata_address_or_partition_label_or_path, fbs::model_location_type_t location = fbs::MODEL_LOCATION_IN_FLASH_RODATA, int model_index = 0, const uint8_t *key = nullptr, bool param_copy = true)

Load model graph and parameters from FLASH or sdcard.

Parameters

- **rodata_address_or_partition_label_or_path** – The address of model data while location is MODEL_LOCATION_IN_FLASH_RODATA. The label of partition while location is MODEL_LOCATION_IN_FLASH_PARTITION. The path of model while location is MODEL_LOCATION_IN_SDCARD.
- **location** – The model location.
- **model_index** – The model index of packed models.
- **key** – The key of encrypted model.
- **param_copy** – Set to false to avoid copy model parameters from FLASH to PSRAM. Only set this param to false when your PSRAM resource is very tight. This saves PSRAM and sacrifices the performance of model inference because the frequency of PSRAM is higher than FLASH. Only takes effect when MODEL_LOCATION_IN_FLASH_RODATA(CONFIG_SPIRAM_RODATA not set) or MODEL_LOCATION_IN_FLASH_PARTITION.

Returns

- ESP_OK Success
- ESP_FAIL Failed

virtual esp_err_t **load**(const char *rodata_address_or_partition_label_or_path, fbs::model_location_type_t location = fbs::MODEL_LOCATION_IN_FLASH_RODATA, const char *model_name = nullptr, const uint8_t *key = nullptr, bool param_copy = true)

Load model graph and parameters from FLASH or sdcard.

Parameters

- **rodata_address_or_partition_label_or_path** – The address of model data while location is MODEL_LOCATION_IN_FLASH_RODATA. The label of partition while location is MODEL_LOCATION_IN_FLASH_PARTITION. The path of model while location is MODEL_LOCATION_IN_SDCARD.
- **location** – The model location.
- **model_name** – The model name of packed models.
- **key** – The key of encrypted model.
- **param_copy** – Set to false to avoid copy model parameters from FLASH to PSRAM. Only set this param to false when your PSRAM resource is very tight. This saves PSRAM and sacrifices the performance of model inference because the frequency of PSRAM is higher than FLASH. Only takes effect when MODEL_LOCATION_IN_FLASH_RODATA(CONFIG_SPIRAM_RODATA not set) or MODEL_LOCATION_IN_FLASH_PARTITION.

Returns

- ESP_OK Success
- ESP_FAIL Failed

virtual esp_err_t **load**(fbs::*FbsModel* *fbs_model)

Load model graph and parameters from Flatbuffers model.

Parameters
fbs_model – The FlatBuffers model

Returns

- ESP_OK Success
- ESP_FAIL Failed

virtual void **build**(size_t max_internal_size, memory_manager_t mm_type = MEMORY_MANAGER_GREEDY, bool preload = false)

Allocate memory for the model.

Parameters

- **max_internal_size** – In bytes. Limit the max internal size usage. Only take effect when there's a PSRAM, and you want to alloc memory on internal RAM first.
- **mm_type** – Type of memory manager
- **preload** – Whether to preload the model's parameters to internal ram (not implemented yet)

virtual void **run**(runtime_mode_t mode = RUNTIME_MODE_SINGLE_CORE)

Run the model module by module.

Parameters
mode – Runtime mode.

virtual void **run**(*TensorBase* *input, runtime_mode_t mode = RUNTIME_MODE_SINGLE_CORE)

Run the model module by module.

Parameters

- **input** – The model input.
- **mode** – Runtime mode.

virtual void **run**(std::map<std::string, *TensorBase**> &user_inputs, runtime_mode_t mode = RUNTIME_MODE_SINGLE_CORE, std::map<std::string, *TensorBase**> user_outputs = {})

Run the model module by module.

Parameters

- **user_inputs** – The model inputs.
- **mode** – Runtime mode.
- **user_outputs** – It's for debug to specify the output of the intermediate layer; Under normal use, there is no need to pass a value to this parameter. If no parameter is passed, the default is the graphical output, which can be obtained through *Model::get_outputs()*.

void **minimize**()

Minimize the model.

esp_err_t **test**()

Test whether the model inference result is correct. The model should contain test_inputs and test_outputs. Enable export_test_values option in esp-ppq to use this api.

Returns
esp_err_t

std::map<std::string, mem_info_t> **get_memory_info**()

Get memory info.

Returns
Memory usage statistics on internal and PSRAM.

std::map<std::string, module_info> **get_module_info**()

Get module info.

Returns
return Type and latency of each module.

void **print_module_info**(const std::map<std::string, module_info> &info, bool sort_module_by_latency = false)

Print the module info obtained by get_module_info function.

Parameters

- **info** –
- **sort_module_by_latency** –

void **profile_memory**()

Print model memory summary.

void **profile_module**(bool sort_module_by_latency = false)

Print module info summary. (Name, Type, Latency)

Parameters
sort_module_by_latency – True The module is printed in latency decreasing sort. False The module is printed in ONNX topological sort.

void **profile**(bool sort_module_by_latency = false)

Combination of profile_memory & profile_module.

Parameters
sort_module_by_latency – True The module is printed in latency decreasing sort. False The module is printed in ONNX topological sort.

virtual std::map<std::string, *TensorBase**> &**get_inputs**()

Get inputs of model.

Returns
The map of model input's name and TensorBase*

virtual *TensorBase* ***get_input**()

Get the only input of model.

Returns
TensorBase*

virtual *TensorBase* ***get_input**(const std::string &name)

Get input of model by name.

Parameters
name – input name

Returns
TensorBase*

virtual *TensorBase* ***get_intermediate**(const std::string &name)

Get intermediate *TensorBase* of model.

Note: When using memory manager, the content of *TensorBase*'s data may be overwritten by the outputs of other

---

Parameters
name – The name of intermediate Tensor. operators.

Returns
The intermediate TensorBase*.

virtual std::map<std::string, *TensorBase**> &**get_outputs**()

Get outputs of model.

Returns
The map of model output's name and TensorBase*

virtual *TensorBase* ***get_output**()

Get the only output of model.

Returns
TensorBase*

virtual *TensorBase* ***get_output**(const std::string &name)

Get output of model by name.

Parameters
name – output name

Returns
TensorBase*

std::string **get_metadata_prop**(const std::string &key)

Get the model's metadata prop.

Parameters
key – The key of metadata prop

Returns
The value of metadata prop

virtual void **print**()

Print the model.

virtual void **reset**()

Reset the model. This function resets the model context and all modules, but does not reload the model or rebuild the execution plan.

inline virtual fbs::*FbsModel* ***get_fbs_model**()

Get the fbs model instance.

Returns
fbs::FbsModel *

4.3.4 Header File

- esp-dl/dl/model/include/dl_model_context.hpp

4.3.5 Macros

CONTEXT_PARAMETER_OFFSET

Offset for parameter tensors

4.3.6 Classes

class **ModelContext**

Model Context class including variable tensors and parameters.

Public Functions

inline **ModelContext**()

Constructor for *ModelContext*. Initializes the PSRAM and internal root pointers to nullptr.

inline **~ModelContext**()

Destructor for *ModelContext*. Clears all resources and tensors.

int **add_tensor**(const std::string name, bool is_paramter = false, *TensorBase* *tensor = nullptr)

Adds a tensor to the parameter or variable list.

Parameters

- **name** – The name of the tensor.
- **is_paramter** – Whether the tensor is a parameter (default: false).
- **tensor** – Pointer to the *TensorBase* object (default: nullptr).

Returns

int Returns the index of the added tensor.

int **push_back_tensor**(*TensorBase* *tensor, bool is_paramter = false)

Push back a tensor.

Parameters

- **tensor** – Pointer to the *TensorBase* object.
- **is_paramter** – Whether the tensor is a parameter (default: false).

Returns

int Returns the index of the added tensor.

void **update_tensor**(int index, *TensorBase* *tensor)

Updates the tensor at the specified index.

Parameters

- **index** – The index of the tensor to update.
- **tensor** – Pointer to the new *TensorBase* object.

TensorBase ***get_tensor**(int index)

Gets the tensor by its index.

Parameters
index – The index of the tensor.

Returns
TensorBase* Returns the pointer to the *TensorBase* object, or nullptr if the index is invalid.

TensorBase ***get_tensor**(const std::string &name)

Gets the tensor by its name.

Parameters
name – The name of the tensor.

Returns
TensorBase* Returns the pointer to the *TensorBase* object, or nullptr if the name is not found.

int **get_tensor_index**(const std::string &name)

Gets the tensor index by its name.

Parameters
name – The name of the tensor.

Returns
int Returns index if the name is found, else -1

int **get_variable_index**(const std::string &name)

Gets the variable tensor index by its name.

Parameters
name – The name of the tensor.

Returns
int Returns index if the name is found and is variable tensor, else -1

inline int **get_variable_count**()

Gets the count of variable tensors.

Returns
int Returns the number of variable tensors.

inline int **get_parameter_count**()

Gets the count of parameter tensors.

Returns
int Returns the number of parameter tensors.

bool **root_alloc**(size_t internal_size, size_t psram_size, int alignment = 16)

Allocates memory for PSRAM and internal roots.

Parameters

- **internal_size** – The size of the internal memory in bytes.
- **psram_size** – The size of the PSRAM memory in bytes.
- **alignment** – The alignment of the memory in bytes.

Returns
Bool Return true if the allocation is successful, false otherwise.

inline void *`get_psram_root`()

Gets the pointer to the PSRAM root.

Returns
Void* Returns the pointer to the PSRAM root.

inline void *`get_internal_root`()

Gets the pointer to the internal root.

Returns
Void* Returns the pointer to the internal root.

size_t `get_parameter_memory_size`(mem_info_t &mem_info, bool copy)

Gets the size of the parameters in bytes.

Parameters

- `mem_info` – The size of the memory used by the parameters in bytes, filtered by copy option.
- `copy` – Filter the parameters by auto_free.

Returns
size_t Returns the total size of the parameters memory in bytes.

size_t `get_variable_memory_size`(mem_info_t &mem_info)

Get the variable memory size object.

Parameters
`mem_info` – The size of the memory used by the variables in bytes.

Returns
size_t Returns the total size of the variables memory in bytes.

inline void `root_free`()

Frees the memory allocated for PSRAM and internal roots. This function ensures proper cleanup of allocated memory.

inline void `root_reset`()

Resets the context by clearing variables, parameters, and name-to-index map. This is used to reset the state of the context for a new inference or after an inference.

inline void `minimize`()

Minimizes the context by clearing the name-to-index map. This is used to free unnecessary intermediate variables during the inference.

inline void `clear`()

Clears all resources and tensors in the context. This includes clearing variables, parameters, name-to-index map, and freeing memory.

Public Members

std::vector<*TensorBase**> **m_variables**

Variable tensors of model, the first one is nullptr

std::vector<*TensorBase**> **m_parameters**

Parameters of model, the first one is nullptr

4.3.7 Header File

- esp-dl/dl/model/include/dl_memory_manager.hpp

4.3.8 Classes

class **MemoryManagerBase**

Memory manager base class, each model has its own memory manager TODO: share memory manager with different models.

Subclassed by *dl::MemoryManagerGreedy*

Public Functions

inline **MemoryManagerBase**(int alignment = 16)

Construct a new Memory Manager Base object.

Parameters

alignment – Memory address alignment

inline virtual **~MemoryManagerBase**()

Destroy the MemoryManager object. Return resource.

virtual bool **alloc**(fbs::*FbsModel* *fbs_model, std::vector<dl::module::*Module**> &execution_plan, *ModelContext* *context) = 0

Allocate memory for each tensor, include all input and output tensors.

Parameters

- **fbs_model** – FlatBuffer's *Model*
- **execution_plan** – Topological sorted module list
- **context** – *Model* context

Returns

Bool Return true if the allocation is successful, false otherwise.

Public Members

int **alignment**

The root pointer needs to be aligned must be a power of two

class **TensorInfo**

Tensor info, include tensor name, shape, dtype, size, time range and call times, which is used to plan model memory.

Public Functions

TensorInfo(std::string &name, int time_begin, int time_end, std::vector<int> shape, dtype_t dtype, int exponent, bool is_internal = false)

Construct a new Tensor Info object.

Parameters

- **name** – Tensor name
- **time_begin** – Tensor lifetime begin
- **time_end** – Tensor lifetime end
- **shape** – Tensor shape
- **dtype** – Tensor dtype
- **exponent** – Tensor exponent
- **is_internal** – Is tensor in internal RAM or not

inline **~TensorInfo**()

Destroy the Tensor Info object.

void **set_inplace_leader_tensor**(*TensorInfo* *tensor)

Set the inplace leader tensor object.

Parameters
tensor – Inplace leader tensor

inline void **set_inplace_follower_dirty_tensor**(*TensorInfo* *tensor)

Set the inplace follower dirty tensor object.

Parameters
tensor – Inplace follower dirty tensor

inline void **set_inplace_follower_clean_tensor**(*TensorInfo* *tensor)

Set the inplace follower clean tensor object.

Parameters
tensor – Inplace follower clean tensor

inline std::pair<*TensorInfo**, *TensorInfo**> **get_inplace_follower_tensor**()

Get the inplace follower tensor object.

Returns
std::pair<TensorInfo *, TensorInfo *>

void **update_time**(int new_time)

Update Tensor lifetime.

Parameters
new_time – new tensor lifetime

TensorBase ***create_tensor**(void *internal_root, void *psram_root)

Create a *TensorBase* object according to *TensorInfo*.

Parameters

- **internal_root** – Internal RAM root pointer
- **psram_root** – PSRAM root pointer

Returns
TensorBase*

inline bool **is_inplaced**()

Is inplaced or not.

Returns
true if inplaced else false

inline uint32_t **get_offset**()

Get the tensor offset.

Returns
uint32_t

inline void **set_offset**(uint32_t offset)

Set the tensor offset.

Parameters
offset –

inline uint32_t **get_internal_offset**()

Get the internal offset.

Returns
uint32_t

inline bool **get_internal_state**()

Get the internal state.

Returns
true if is internal else false

inline void **set_internal_state**(bool is_internal)

Set the internal state.

Parameters
is_internal –

inline void **set_internal_offset**(uint32_t offset)

Set the internal offset.

Parameters
offset –

inline int **get_time_end**()

Get the liftetime end.

Returns
int

inline int **get_time_begin**()

Get the liftetime begin.

Returns
int

inline size_t **get_size**()

Get the tensor size.

Returns
size_t

inline std::string **get_name**()

Get the tensor name.

Returns
std::string

inline std::vector<int> **get_shape**()

Get the tensor shape.

Returns
std::vector<int>

inline void **print**()

print tensor info

class **MemoryChunk**

Memory chunk, include size, is free, offset, alignment and tensor, which is used to simulate memory allocation.

Public Functions

MemoryChunk(size_t size, int is_free, int alignment = 16)

Construct a new Memory Chunk object.

Parameters

- **size** – Memory chunk size
- **is_free** – Whether free or not
- **alignment** – Memory chunk alignment

MemoryChunk(*TensorInfo* *tensor, int alignment = 16)

Construct a new Memory Chunk object.

Parameters

- **tensor** – *TensorInfo*
- **alignment** – Memory chunk alignment

inline **~MemoryChunk**()

Destroy the Memory Chunk object.

MemoryChunk ***merge_free_chunk**(*MemoryChunk* *chunk)

Merge continuous free chunk.

Parameters
chunk –

Returns
MemoryChunk*

MemoryChunk ***insert**(*TensorInfo* *tensor)

Insert tensor into free chunk.

Parameters
tensor –

Returns
MemoryChunk*

MemoryChunk ***extend**(*TensorInfo* *tensor)

Extend free chunk and insert tensor.

Parameters
tensor –

Returns
MemoryChunk*

inline void **free**()

Free memory chunk, set is_free to true and set tensor to nullptr.

Public Members

size_t **size**

Memory chunk size

bool **is_free**

Whether memory chunk is free or not

int **offset**

Offset relative to root pointer

int **alignment**

Memory address alignment

TensorInfo ***tensor**

Info of the tensor which occupies the memory

4.3.9 Header File

- esp-dl/dl/model/include/dl_memory_manager_greedy.hpp

4.3.10 Classes

class **MemoryManagerGreedy** : public dl::*MemoryManagerBase*

Greedy memory manager that allocates memory for tensors in execution order, prioritizing internal RAM allocation first.

Public Functions

inline **MemoryManagerGreedy**(int max_internal_size, int alignment = 16)

Constructs a greedy memory manager with specified constraints.

Parameters

- **max_internal_size** – Maximum allowed internal RAM usage in bytes
- **alignment** – Memory address alignment requirement (default: 16 bytes)

inline **~MemoryManagerGreedy**()

Destructor that releases all managed memory resources.

virtual bool **alloc**(fbs::*FbsModel* *fbs_model, std::vector<dl::module::*Module**> &execution_plan, *ModelContext* *context)

Allocates memory for all network tensors following greedy strategy.

Parameters

- **fbs_model** – FlatBuffer model containing network architecture
- **execution_plan** – Execution graph ordered by computation dependencies
- **context** – Device-specific runtime configuration

Returns

bool True if successful allocation, false if memory insufficient

void **free**()

Releases all allocated memory including tensor buffers and memory pools.

4.4 Fbs API Reference

The esp-dl model utilizes FlatBuffers to store information about parameters and the computation graph. Taking into account the encryption requirements of some models, this part has not been open-sourced. However, we provide a set of APIs to facilitate users in loading and parsing esp-dl models.

4.4.1 Header File

- esp-dl/fbs_loader/include/fbs_loader.hpp

4.4.2 Classes

class **FbsLoader**

Class for parser the flatbuffers.

Public Functions

FbsLoader(const char *rodata_address_or_partition_label_or_path = nullptr, model_location_type_t location = MODEL_LOCATION_IN_FLASH_RODATA)

Construct a new *FbsLoader* object.

Parameters

- **rodata_address_or_partition_label_or_path** – The address of model data while location is MODEL_LOCATION_IN_FLASH_RODATA. The label of partition while location is MODEL_LOCATION_IN_FLASH_PARTITION. The path of model while location is MODEL_LOCATION_IN_SDCARD.
- **location** – The model location.

~FbsLoader()

Destroy the *FbsLoader* object.

FbsModel ***load**(const uint8_t *key = nullptr, bool param_copy = true)

Load the model. If there are multiple sub-models, the first sub-model will be loaded.

Parameters

- **key** – NULL or a 128-bit AES key, like {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f}
- **param_copy** – Set to false to avoid copy model parameters from FLASH to PSRAM. Only set this param to false when your PSRAM resource is very tight. This saves PSRAM and sacrifices the performance of model inference because the frequency of PSRAM is higher than FLASH. Only takes effect when MODEL_LOCATION_IN_FLASH_RODATA(CONFIG_SPIRAM_RODATA not set) or MODEL_LOCATION_IN_FLASH_PARTITION.

Returns

Return nullptr if loading fails. Otherwise return the pointer of *FbsModel*.

FbsModel ***load**(const int model_index, const uint8_t *key = nullptr, bool param_copy = true)

Load the model by model index.

Parameters

- **model_index** – The index of model.
- **key** – NULL or a 128-bit AES key, like {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f}.

- **param_copy** – Set to false to avoid copy model parameters from FLASH to PSRAM. Only set this param to false when your PSRAM resource is very tight. This saves PSRAM and sacrifices the performance of model inference because the frequency of PSRAM is higher than FLASH. Only takes effect when MODEL_LOCATION_IN_FLASH_RODATA(CONFIG_SPIRAM_RODATA not set) or MODEL_LOCATION_IN_FLASH_PARTITION.

Returns
Return nullptr if loading fails. Otherwise return the pointer of *FbsModel*.

FbsModel ***load**(const char *model_name, const uint8_t *key = nullptr, bool param_copy = true)

Load the model by model name.

Parameters

- **model_name** – The name of model.
- **key** – NULL or a 128-bit AES key, like {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f}
- **param_copy** – Set to false to avoid copy model parameters from FLASH to PSRAM. Only set this param to false when your PSRAM resource is very tight. This saves PSRAM and sacrifices the performance of model inference because the frequency of PSRAM is higher than FLASH. Only takes effect when MODEL_LOCATION_IN_FLASH_RODATA(CONFIG_SPIRAM_RODATA not set) or MODEL_LOCATION_IN_FLASH_PARTITION.

Returns
Return nullptr if loading fails. Otherwise return the pointer of *FbsModel*.

int **get_model_num**()

Get the number of models.

Returns
The number of models

void **list_models**()

List all model's name.

const char ***get_model_location_string**()

Get the model location string.

Returns
The model location string.

4.4.3 Header File

- esp-dl/fbs_loader/include/fbs_model.hpp

4.4.4 Classes

class **FbsModel**

Flatbuffer model object.

Public Functions

FbsModel(const void *data, size_t size, model_location_type_t location, bool encrypt, bool rodata_move, bool auto_free, bool param_copy)

Construct a new *FbsModel* object.

Parameters

- **data** – The data of model flatbuffers.
- **size** – The size of model flatbuffers in bytes.
- **location** – The location of model flatbuffers.
- **encrypt** – Whether the model flatbuffers is encrypted or not.
- **rodata_move** – Whether the model flatbuffers is moved from FLASH rodata to PSRAM.
- **auto_free** – Whether to free the model flatbuffers data when destroy this class instance.
- **param_copy** – Whether to copy the parameter in flatbuffers.

~FbsModel()

Destroy the *FbsModel* object.

void **print**()

Print the model information.

std::vector<std::string> **topological_sort**()

Return vector of node name in the order of execution.

Returns

topological sort of node name.

esp_err_t **get_operation_attribute**(std::string node_name, std::string attribute_name, int &ret_value)

Get the attribute of node.

Parameters

- **node_name** – The name of operation.
- **attribute_name** – The name of attribute.
- **ret_value** – The attribute value.

Returns

esp_err_t Return ESP_OK if get successfully. Otherwise return ESP_FAIL.

esp_err_t **get_operation_attribute**(std::string node_name, std::string attribute_name, float &ret_value)

Get the attribute of node.

Parameters

- **node_name** – The name of operation.
- **attribute_name** – The name of attribute.

- **ret_value** – The attribute value.

Returns
esp_err_t Return ESP_OK if get successfully. Otherwise return ESP_FAIL.

esp_err_t **get_operation_attribute**(std::string node_name, std::string attribute_name, std::string &ret_value)

Get the attribute of node.

Parameters

- **node_name** – The name of operation.
- **attribute_name** – The name of attribute.
- **ret_value** – The attribute value.

Returns
esp_err_t Return ESP_OK if get successfully. Otherwise return ESP_FAIL.

esp_err_t **get_operation_attribute**(std::string node_name, std::string attribute_name, std::vector<int> &ret_value)

Get the attribute of node.

Parameters

- **node_name** – The name of operation.
- **attribute_name** – The name of attribute.
- **ret_value** – The attribute value.

Returns
esp_err_t Return ESP_OK if get successfully. Otherwise return ESP_FAIL.

esp_err_t **get_operation_attribute**(std::string node_name, std::string attribute_name, std::vector<float> &ret_value)

Get the attribute of node.

Parameters

- **node_name** – The name of operation.
- **attribute_name** – The name of attribute.
- **ret_value** – The attribute value.

Returns
esp_err_t Return ESP_OK if get successfully. Otherwise return ESP_FAIL.

esp_err_t **get_operation_attribute**(std::string node_name, std::string attribute_name, dl::quant_type_t &ret_value)

Get the attribute of node.

Parameters

- **node_name** – The name of operation.
- **attribute_name** – The name of attribute.
- **ret_value** – The attribute value.

Returns
esp_err_t Return ESP_OK if get successfully. Otherwise return ESP_FAIL.

esp_err_t **get_operation_attribute**(std::string node_name, std::string attribute_name, dl::activation_type_t &ret_value)

Get the attribute of node.

Parameters

- **node_name** – The name of operation.
- **attribute_name** – The name of attribute.
- **ret_value** – The attribute value.

Returns
esp_err_t Return ESP_OK if get successfully. Otherwise return ESP_FAIL.

esp_err_t **get_operation_attribute**(std::string node_name, std::string attribute_name, dl::resize_mode_t &ret_value)

Get the attribute of node.

Parameters

- **node_name** – The name of operation.
- **attribute_name** – The name of attribute.
- **ret_value** – The attribute value.

Returns
esp_err_t Return ESP_OK if get successfully. Otherwise return ESP_FAIL.

esp_err_t **get_operation_attribute**(std::string node_name, std::string attribute_name, dl::*TensorBase* *&ret_value)

Get the attribute of node.

Parameters

- **node_name** – The name of operation.
- **attribute_name** – The name of attribute.
- **ret_value** – The attribute value.

Returns
esp_err_t Return ESP_OK if get successfully. Otherwise return ESP_FAIL.

esp_err_t **get_operation_input_shape**(std::string node_name, int index, std::vector<int> &ret_value)

Get operation input shape.

Parameters

- **node_name** – The name of operation.
- **index** – The index of inputs
- **ret_value** – Return shape value.

Returns
esp_err_t Return ESP_OK if get successfully. Otherwise return ESP_FAIL.

esp_err_t **get_operation_output_shape**(std::string node_name, int index, std::vector<int> &ret_value)

Get operation output shape.

Parameters

- **node_name** – The name of operation.
- **index** – The index of outputs
- **ret_value** – Return shape value.

Returns
esp_err_t Return ESP_OK if get successfully. Otherwise return ESP_FAIL.

esp_err_t **get_operation_inputs_and_outputs**(std::string node_name, std::vector<std::string> &inputs, std::vector<std::string> &outputs)

Get the attribute of node.

Parameters

- **node_name** – The name of operation.
- **inputs** – The vector of operation inputs.
- **outputs** – The vector of operation outputs.

Returns
esp_err_t Return ESP_OK if get successfully. Otherwise return ESP_FAIL.

std::string **get_operation_type**(std::string node_name)

Get operation type, “Conv”, “Linear” etc.

Parameters
node_name – The name of operation

Returns
The type of operation.

dl::*TensorBase* ***get_operation_parameter**(std::string node_name, int index = 1, uint32_t caps = MALLOC_CAP_DEFAULT)

Return if the variable is a parameter.

Parameters

- **node_name** – The name of operation
- **index** – The index of the variable
- **caps** – Bitwise OR of MALLOC_CAP_* flags indicating the type of memory to be returned

Returns
*dl::TensorBase**

dl::*TensorBase* ***get_operation_lut**(std::string node_name, uint32_t caps = MALLOC_CAP_DEFAULT, std::string attribute_name = "lut")

Get LUT(Look Up Table) if the operation has LUT.

Parameters

- **node_name** – The name of operation
- **caps** – Bitwise OR of MALLOC_CAP_* flags indicating the type of memory to be returned
- **attribute_name** – The name of LUT attribute

Returns
*dl::TensorBase**

bool **is_parameter**(std::string name)

return true if the variable is a parameter

Parameters
name – Variable name

Returns
true if the variable is a parameter else false

const void ***get_tensor_raw_data**(std::string tensor_name)

Get the raw data of FlatBuffers::Dl::Tensor.

Parameters
tensor_name – The name of Tensor.

Returns
uint8_t * The pointer of raw data.

dl::dtype_t **get_tensor_dtype**(std::string tensor_name)

Get the element type of tensor tensor.

Parameters
tensor_name – The tensor name.

Returns
FlatBuffers::Dl::TensorDataType

std::vector<int> **get_tensor_shape**(std::string tensor_name)

Get the shape of tensor.

Parameters
tensor_name – The name of tensor.

Returns
std::vector<int> The shape of tensor.

std::vector<int> **get_tensor_exponents**(std::string tensor_name)

Get the exponents of tensor.

> **Warning:** When quantization is PER_CHANNEL, the size of exponents is same as out_channels. When quantization is PER_TENSOR, the size of exponents is 1.

Parameters
tensor_name – The name of tensor.

Returns
The exponents of tensor.

dl::dtype_t **get_value_info_dtype**(std::string var_name)

Get the element type of value_info.

Parameters
var_name – The value_info name.

Returns
dl::dtype_t

std::vector<int> **get_value_info_shape**(std::string var_name)

Get the shape of value_info.

Parameters
var_name – The value_info name.

Returns
the shape of value_info.

int **get_value_info_exponent**(std::string var_name)

Get the exponent of value_info. Only support PER_TENSOR quantization.

Parameters
var_name – The value_info name.

Returns
the exponent of value_info

const void ***get_test_input_tensor_raw_data**(std::string tensor_name)

Get the raw data of test input tensor.

Parameters
tensor_name – The name of test input tensor.

Returns
uint8_t * The pointer of raw data.

const void ***get_test_output_tensor_raw_data**(std::string tensor_name)

Get the raw data of test output tensor.

Parameters
tensor_name – The name of test output tensor.

Returns
uint8_t * The pointer of raw data.

dl::*TensorBase* ***get_test_input_tensor**(std::string tensor_name)

Get the test input tensor.

Parameters
tensor_name – The name of test input tensor.

Returns
The pointer of tensor.

dl::*TensorBase* ***get_test_output_tensor**(std::string tensor_name)

Get the test output tensor.

Parameters
tensor_name – The name of test output tensor.

Returns
The pointer of tensor.

std::vector<std::string> **get_test_outputs_name**()

Get the name of test outputs.

Returns
the name of test outputs

std::vector<std::string> **get_graph_inputs**()

Get the graph inputs.

Returns
the name of inputs

std::vector<std::string> **get_graph_outputs**()

Get the graph outputs.

Returns
the name of ounputs

void **clear_map**()

Clear all map.

void **load_map**()

Load all map.

std::string **get_model_name**()

Get the model name.

Returns
the name of model

int64_t **get_model_version**()

Get the model version.

Returns
The version of model

std::string **get_model_doc_string**()

Get the model doc string.

Returns
The doc string of model

std::string **get_model_metadata_prop**(const std::string &key)

Get the model's metadata prop.

Parameters
key – The key of metadata prop

Returns
The value of metadata prop

void **get_model_size**(size_t *internal_size, size_t *psram_size, size_t *psram_rodata_size, size_t *flash_size)

Get the model size.

Parameters

- **internal_size** – Flatbuffers model internal RAM usage
- **psram_size** – Flatbuffers model PSRAM usage
- **psram_rodata_size** – Flatbuffers model PSRAM rodate usage. If CONFIG_SPIRAM_RODATA option is on, \ Flatbuffers model in FLASH rodata will be copied to PSRAM
- **flash_size** – Flatbuffers model FLASH usage

Public Members

bool **m_param_copy**

copy flatbuffers param or not.