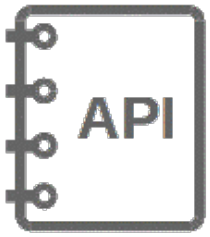
ESP-DL 用户指南

Release v3.2.1-12-g4e13e638db
乐鑫信息科技
2025 年 12 月 23 日

Table of contents

索引 **85**

		
入门指南	使用教程	API Reference

Chapter 1

Introduction

1.1 ESP-DL 简介

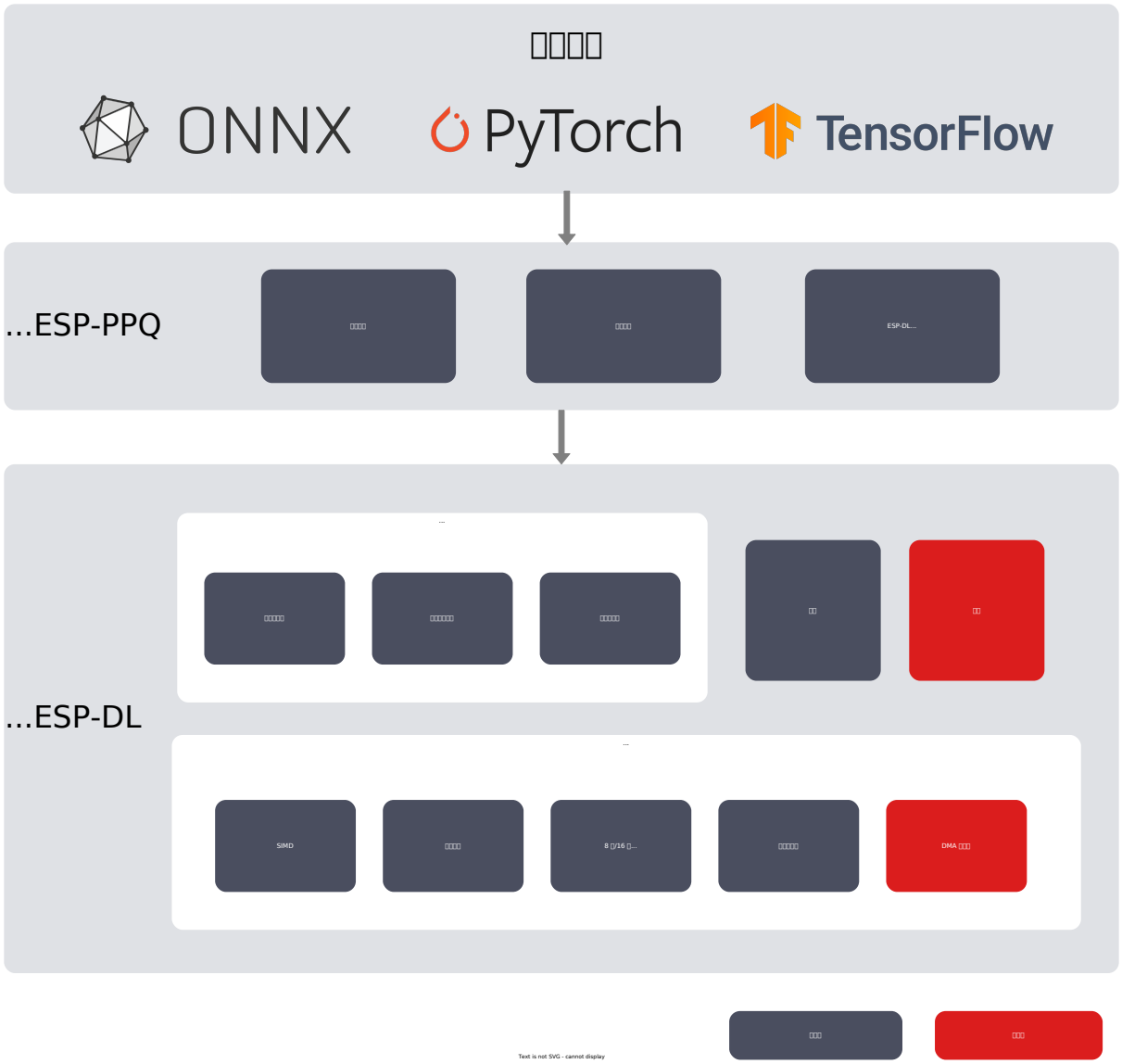
ESP-DL 是一个专为 ESP 系列芯片设计的轻量级且高效的神经网络推理框架。通过 ESP-DL，您可以轻松快速地使用乐鑫的系统级芯片 (SoC) 开发 AI 应用。

1.1.1 概述

ESP-DL 提供了加载、调试和运行 AI 模型的 API。该框架易于使用，并且可以与其他乐鑫 SDK 无缝集成。ESP-PPQ 作为 ESP-DL 的量化工具，能够量化来自 ONNX、Pytorch 和 TensorFlow 的模型，并将其导出为 ESP-DL 标准模型格式。

- **ESP-DL 标准模型格式**：该格式类似于 ONNX，但使用 FlatBuffers 而不是 Protobuf，使其更轻量级并支持零拷贝反序列化，文件后缀为 '.espdl'。
- **高效算子实现：** ESP-DL 高效地实现了常见的 AI 算子，如 Conv、Pool、Gemm、Add 和 Mul 等。目前支持的算子 operator_support_state.md
- **静态内存规划器：**内存规划器根据用户指定的内部 RAM 大小，自动将不同层分配到最佳内存位置，确保高效的整体运行速度同时最小化内存使用。
- **双核调度：**自动双核调度允许计算密集型算子充分利用双核计算能力。目前，Conv2D 和 DepthwiseConv2D 支持双核调度。
- **8bit LUT Activation：**除了 Relu, PRelu(n>1) 之外的所有激活函数，ESP-DL 默认使用 8bit LUT(Look Up Table) 方式实现, 以加速推理。

ESP-DL 系统框架图如下所示:



1.2 ESP-DL 项目组织

ESP-DL 的模块化设计使其开发、维护和扩展变得高效。项目的组织结构如下：

1.2.1 dl（深度学习）

核心深度学习模块和工具，分为子模块：

- **model** 加载、管理和分配深度学习模型的内存。包含 `dl_model_base` 和 `dl_memory_manager`。
- **module** 算子接口（卷积、池化、激活）。文件：`dl_module_conv`，`dl_module_pool`，`dl_module_relu`。
- **base** 具体的算子实现，包括对芯片（esp32s3/esp32p4）的汇编加速。
- **math** 数学操作（矩阵函数）。文件：`dl_math.hpp` 和 `dl_math.cpp`。
- **tool** 辅助功能（实用工具）。文件：`dl_tool.hpp` 和 `dl_tool.cpp`。
- **tensor** 张量类。文件：`dl_tensor_base.hpp`。

1.2.2 vision（计算机视觉）

计算机视觉模块，分为子模块：

- **classification** 图像分类（模型推理）。推理：`dl_cls_base`。后处理器：`imagenet_cls_postprocessor`，`dl_cls_postprocessor`。
- **recognition** 特征提取（模型推理）。特征数据库管理（注册、删除、查询）。预处理器：`dl_feat_image_preprocessor`。推理：`dl_feat_base`。后处理器：`dl_feat_postprocessor`。数据库：`dl_recognition_database`
- **image** 图像处理（调整大小、裁剪、仿射变换）。颜色转换（像素、图像）。图像预处理器（调整大小、裁剪、颜色转换、规范化、量化的管道）。图像解码/编码（JPEG/BMP）。绘制工具（点、空心矩形）。图像处理：`dl_image_process`。颜色转换：`dl_image_color`。图像预处理器：`dl_image_preprocessor`。图像解码/编码：`dl_image_jpeg`、`dl_image_bmp`。绘制工具：`dl_image_draw`。
- **detect** 目标检测（模型推理）。推理：`dl_detect_base`。后处理器：`dl_detect_msr_postprocessor`、`dl_detect_mnp_postprocessor`、`dl_detect_pico_postprocessor`。

1.2.3 fbs_loader（FlatBuffers 加载器）

处理 FlatBuffers 模型：

- **include** 头文件：`fbs_loader.hpp`，`fbs_model.hpp`。
- **src** 实现：`fbs_loader.cpp`。

1.2.4 其他文件

- **CMakeLists.txt** 项目构建配置。
- **idf_component.yml** 组件元数据（名称、版本、依赖项）。
- **README.md** 项目文档和使用说明。
- **LICENSE** 许可条款。

Chapter 2

入门指南

2.1 硬件要求

- 一块 ESP32-S3 或 ESP32-P4 开发板。推荐使用：ESP32-S3-EYE 或 ESP32-P4-Function-EV-Board
- 一台 PC（Linux 系统）

备注:

- 部分开发板目前采用 Type C 接口。请确保使用正确的线缆连接开发板！
- ESP-DL 也支持 ESP32，但其算子实现采用 C 编写，因此 ESP32 运行速度会远慢于 ESP32-S3 或 ESP32-P4。如有需要，可在项目中自行添加编译配置文件，ESP-DL 的函数接口调用方式完全一致。需要注意的是:
  - 使用 **ESP-PPQ** 量化 **ESP32** 平台模型时，需将 target 设置为 `c`。
  - 使用 **ESP-DL** 部署 **ESP32** 平台模型时，项目编译 target 则设置为 `esp32`。

2.2 软件要求

2.2.1 ESP-IDF

ESP-DL 基于 ESP-IDF 运行。有关如何获取 ESP-IDF 的详细说明，请参阅 ESP-IDF 编程指南。

备注: 请使用 ESP-IDF 的 `release/v5.3` 或更高版本。

2.2.2 ESP-PPQ

ESP-PPQ 是基于 ppq 的量化工具，其 代码 已全部开源。ESP-PPQ 在 PPQ 的基础上添加了乐鑫定制的 quantizer 和 exporter，方便用户根据不同的芯片选择和 ESP-DL 匹配的量化规则，并导出为 ESP-DL 可以直接加载的标准模型文件。ESP-PPQ 兼容 PPQ 所有的 API 和量化脚本。更多细节请参考 PPQ 文档和视频。如果您想量化自己的模型，可以使用如下方式安装 esp-ppq：

方式一: 使用 pip 安装包

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

方式二: 使用 pip 安装源码，以便保持与 master 分支同步

```
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 .
```

方式三: 使用 uv 安装包

```
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
```

方式四: 使用 uv 安装源码，以便保持与 master 分支同步

```
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 .
```

方式五: 在 docker 中使用 esp-ppq

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

备注:

- 示例代码中安装的是 linux pytorch cpu 版本，请根据实际情况安装对应的 pytorch。
- 如果使用 uv 安装包，仅需要更改 `--torch-backend` 参数即可，其会忽略项目中配置的 pytorch URLs 索引。

2.3 快速开始

ESP-DL 提供了一些开箱即用的 示例

2.3.1 示例编译 & 烧录

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

使用具体的芯片替换 `[Soc]`，目前支持 `esp32s3` 和 `esp32p4`。示例暂未添加 `esp32` 的模型和编译配置文件。

2.3.2 示例配置

```
idf.py menuconfig
```

一些示例包含可配置的选项，可以在使用 `idf.py set-target` 指定芯片之后使用 `idf.py menuconfig` 进行配置。

2.3.3 故障排除

查看 ESP-IDF 文档

请参阅 ESP-IDF DOC

擦除 FLASH 和清除示例

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

删除 `build/`、`sdkconfig`、`dependencies.lock`、`managed_components/` 并重试。

2.4 模型量化

首先，请参考 ESP-DL 算子支持状态 operator_support_state.md，确保您的模型中的算子已经得到支持。

ESP-DL 必须使用专有格式 `.espdl` 进行模型部署，深度学习模型需要进行量化和格式转换之后才能使用。ESP-PPQ 提供了 `espdl_quantize_onnx` 和 `espdl_quantize_torch` 两种接口以支持 ONNX 模型和 PyTorch 模型导出为 `.espdl` 模型。其他深度学习框架，如 TensorfFlow, PaddlePaddle 等都需要先将模型转换为 ONNX 。因此请确保您的模型可以转换为 ONNX 模型。更多详细信息，请参阅：

- 如何量化模型
- 如何量化 *MobileNetV2*
- 如何量化 *YOLO11n*
- 如何量化 *YOLO11n-pose*
- 如何量化流式模型

2.5 模型部署

ESP-DL 提供了一系列 API 来快速加载和运行模型。更多详细信息，请参阅：

- 如何加载和测试模型
- 如何进行模型推理
- 如何部署流式模型

Chapter 3

Tutorials

3.1 如何量化模型

ESP-DL 必须使用专有格式 `.espdl` 进行模型部署。这是一种量化模型格式，支持 8bit 和 16bit。在本教程中，我们将以 quantize_sin_model 为例，介绍如何使用 ESP-PPQ 量化并导出 `.espdl` 模型，量化方法为 Post Training Quantization (PTQ)。

3.1.1 准备工作

安装 ESP_PPQ

3.1.2 预训练模型

```
python sin_model.py
```

执行 sin_model.py 。该脚本会训练一个简单的 Pytorch 模型用于拟合 [0, 2pi] 范围内的 sin 函数。训练结束会保存相应的.pth 权重，并导出 ONNX 模型。

备注： ESP-PPQ 提供了 `espdl_quantize_onnx` 和 `espdl_quantize_torch` 两种接口以支持 ONNX 模型和 PyTorch 模型。其他深度学习框架，如 TensorfFlow, PaddlePaddle 等都需要先将模型转换为 ONNX 。

- TensorFlow 转 ONNX tf2onnx
- TFLite 转 ONNX tflite2onnx
- TFLite 转 TensorFlow tflite2tensorflow

- PaddlePaddle 转 ONNX paddle2onnx

3.1.3 量化并导出 .espdl

参考 quantize_torch_model.py 和 quantize_onnx_model.py，了解如何使用 espdl_quantize_onnx 和 espdl_quantize_torch 接口量化并导出 .espdl 模型。

执行脚本后会导出三个文件，分别是：

- **.espdl：ESPDL 模型二进制文件，可以直接用于芯片的推理。
- **.info：ESPDL 模型文本文件，用于调试和确定 .espdl 模型是否被正确导出。包含了模型结构，量化完的模型权重，测试输入/输出等信息。
- **.json：量化信息文件，用于量化信息的保存和加载。

备注：

1. 不同平台的 .espdl 模型不能混用，推理结果会有误差。
   - ESP32 使用的 ROUND 策略是 ROUND_HALF_UP。
     - 使用 **ESP-PPQ** 量化 **ESP32** 平台模型时，需将 target 设置为 c，因为在 ESP-DL 中，其算子实现采用 C 语言编写。
     - 使用 **ESP-DL** 部署 **ESP32** 平台模型时，项目编译 target 则设置为 esp32。
   - ESP32S3 使用的 ROUND 策略是 ROUND_HALF_UP。
   - ESP32P4 使用的则是 ROUND_HALF_EVEN。
2. 目前 ESP-DL 使用的量化策略是对称量化 + POWER OF TWO。

添加测试输入/输出

验证模型在板端的推理结果是否正确，首先需要记录 PC 端的一组测试输入/输出。开启 api 中的 export_test_values 选项，就能将一组测试输入/输出固化在 .espdl 模型中。input_shape 参数和 inputs 参数必须指定其中的一个，input_shape 参数使用随机的测试输入，inputs 则可以指定一个特定的测试输入。.info 文件中可以查看测试输入/输出的值。搜索 test inputs value 和 test outputs value 查看它们。

量化模型推理 & 精度评估

espdl_quantize_onnx 和 espdl_quantize_torch API 会返回 BaseGraph。使用 BaseGraph 构建相应的 TorchExecutor 就可以在 PC 端使用量化模型进行推理了。

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

量化模型推理得到的输出可以用来计算各种精度指标。由于 esp-dl 板端推理的结果是能和 esp-ppq 对齐的，可以直接用该指标评估量化完模型的性能。

备注：

1. 当前 esp-dl 仅支持 batch_size 为 1，不支持多 batch 或者动态 batch。
2. .info 文件中的测试输入/输出，以及量化完的模型权重都是 16 字节对齐的，也就是说如果不满 16 字节，会在后面填充 0。

3.1.4 高级量化方法

如果你的模型使用默认的 8bit 量化方法无法达到满意的结果，我们也提供了如下量化方法可以进一步减少量化模型的性能损失：

训练后量化 (PTQ)

- 混合精度量化
- 层间均衡量化
- 算子分裂量化

量化感知训练 (QAT)

- *YOLO11n* 量化感知训练
- *YOLO11n-pose* 量化感知训练

3.2 如何加载、测试和性能分析模型

在本教程中，我们将介绍如何加载、测试和分析一个 espdl 模型。参考例程

- 准备工作
- 从 `rodata` 中加载模型
- 从 `partition` 中加载模型
- 从 `sdcard` 中加载模型
- 测试模型板端推理是否正确
- 分析模型内存使用情况
- 分析模型推理延迟
- 组合性能分析：*profile() 方法*

3.2.1 准备工作

1. 安装 *ESP_IDF*
2. 量化导出 *espdl* 模型

3.2.2 从 `rodata` 中加载模型

此方法将模型文件直接嵌入到应用程序 FLASH 的 `.rodata` 段中。这是最简单的方法，但缺点是每次应用程序代码更改时模型都会被重新烧录。

1. **在 `CMakeLists.txt` 中添加模型文件**
   要将 `.espdl` 模型文件嵌入到 `.rodata` 段，请在 `CMakeLists.txt` 中添加以下代码。前几行应放在 `idf_component_register()` 之前，最后一行放在 `idf_component_register()` 之后。

```
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)
```

(下页继续)

(续上页)

```
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. **在程序中加载模型**
   包含头文件：

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

   声明模型符号并创建模型：

```
// 符号名由三部分组成：前缀 "_binary_"，文件名 "model_espdl"，后缀 "_start"
extern const uint8_t model_espdl[] asm("_binary_model_espdl_start");

// 基本用法 - 使用默认参数加载模型
dl::Model *model = new dl::Model((const char *)model_espdl, fbs::MODEL_
↪LOCATION_IN_FLASH_RODATA);

// 高级用法 - 自定义参数：
// - 将参数保留在 FLASH 中（节省 PSRAM/内部 RAM，但性能较低）
// - 限制内部 RAM 使用为 0 字节（优先使用 PSRAM）
// - 使用贪婪内存管理器
// - 无加密密钥
// - param_copy = false（将参数保留在 FLASH 中）
// 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
```

备注：性能与内存权衡：

- **烧录时间：** 使用从 *rodata* 中加载模型 时，模型文件嵌入在应用程序二进制文件中，每次修改代码时都会重新烧录。对于大型模型，这会增加烧录时间。考虑使用从 *partition* 中加载模型 或从 *sdcard* 中加载模型 来避免此问题。
- **内存 vs 性能：** `param_copy` 参数控制模型参数是否从 FLASH 复制到更快的内存（PSRAM/内部 RAM）。设置 `param_copy=false` 可以节省 RAM，但由于 FLASH 访问速度较慢，会降低推理性能。仅在 RAM 极其紧张时才禁用参数复制。
- **应用程序分区大小：** 嵌入在 `.rodata` 中的大型模型可能需要增加 `partition.csv` 中的应用程序分区大小。

3.2.3 从 `partition` 中加载模型

此方法将模型存储在单独的 FLASH 分区中，允许您独立于应用程序代码更新模型。

1. **在 `partition.csv` 中添加模型信息**
   创建或修改您的 `partition.csv` 文件以包含模型分区。有关分区表的详细信息，请参阅 ESP-IDF 分区表文档。

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

   - **Name:** 任何有意义的名称（包括空终止符最多 16 个字符）
   - **Type:** `data`

- **SubType:** spiffs （模型存储必需）
- **Offset:** 留空以自动计算
- **Size:** 必须大于模型文件大小

2. **在 CMakeLists.txt 中添加模型烧录信息**

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

esptool_py_flash_to_partition 中的第二个参数必须与 partition.csv 中的 Name 字段匹配。

3. **在程序中加载模型**
包含头文件：

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

创建模型实例：

```
// 基本用法 - 使用默认参数加载模型
dl::Model *model = new dl::Model("model", fbs::MODEL_LOCATION_IN_FLASH_
↪PARTITION);

// 高级用法 - 将参数保留在 FLASH 中以节省 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
```

第一个参数（分区标签）必须与 partition.csv 中的 Name 字段匹配。

备注：烧录优化：使用 idf.py app-flash 代替 idf.py flash，可以仅烧录应用程序分区而不重新烧录模型分区。这显著减少了开发期间的烧录时间。

3.2.4 从 sdcard 中加载模型

此方法从 SD 卡加载模型，当 FLASH 空间有限或需要频繁更新模型而无需重新烧录时非常有用。

1. **准备 SD 卡**
   - **格式：** SD 卡应格式化为 FAT32。如果未格式化，挂载时将自动格式化（数据会丢失）。
   - **备份：**在使用 ESP-DL 之前，请始终备份 SD 卡数据。
2. **挂载 SD 卡**
   - **使用 BSP（板级支持包）：**
     在 menuconfig 中启用 CONFIG_BSP_SD_FORMAT_ON_MOUNT_FAIL 以允许自动格式化。

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

   - **不使用 BSP：**
     配置挂载选项，设置 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
     };
     // 挂载 SD 卡（具体实现取决于您的硬件）
     ```

3. **复制模型到 SD 卡**
   将您的 `.espdl` 模型文件复制到 SD 卡（例如，复制到根目录作为 `model.espdl`）。
4. **在程序中加载模型**
   包含头文件：

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

- 如果不使用 BSP(Board Support Package)

```
// 挂载sdcard.
const char *model_path = "/your_sdcard_mount_point/your_model_path/model_name.
↪espdl";
Model *model = new Model(model_path, fbs::MODEL_LOCATION_IN_SDCARD);
```

备注： 使用*从 sdcard 中加载模型* 时，模型加载过程将花费更长的时间，因为模型数据需要从 sdcard 复制到 PSRAM 或者 internal RAM。如果你的 FLASH 空间紧张，这种方法很有用。

3.2.5 测试模型板端推理是否正确

`test()` 方法通过将推理结果与模型文件中嵌入的基准真值进行比较，验证模型是否产生正确的推理结果。

前提条件：

- `.espdl` 模型必须在 ESP-PPQ 中导出时启用 ** 测试输入和输出 **（使用 `export_test_values` 选项）。
- 对于部署，您可以导出一个没有测试数据的版本以减小模型大小。

API： `esp_err_t dl::Model::test()`

返回值： 如果所有测试通过则返回 `ESP_OK`，否则返回 `ESP_FAIL`。

用法：

```
#include "dl_model_base.hpp"

// 创建模型后...
esp_err_t ret = model->test();
if (ret == ESP_OK) {
    ESP_LOGI(TAG, "模型测试通过！");
} else {
    ESP_LOGE(TAG, "模型测试失败！");
}

// 或使用便捷宏：
ESP_ERROR_CHECK(model->test());
```

工作原理：

1. 加载模型中嵌入的测试输入张量，所以 test() 不需要外部输入
2. 通过所有模型层运行推理
3. 将每个输出与基准真值进行比较（考虑量化误差的容差）
4. 报告每个输出的成功或失败

INT16 模型注意事项： 由于量化舍入误差，INT16 模型允许比较时有 ±1 的差异。

3.2.6 分析模型内存使用情况

`profile_memory()` 方法打印跨不同内存类型（内部 RAM、PSRAM、FLASH）的内存使用详细明细。

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

用法:

```
#include "dl_model_base.hpp"

// 创建并测试模型后...
model->profile_memory();
```

输出包括:

名称	解释
fbs_model parameter	flatbuffers 模型，包含一个子项，模型参数 parameter。flatbuffers 模型除了模型参数之外，还包括测试输入输出，模型参数/变量的形状，模型结构等信息。
parameter_copy	复制的模型参数，当 flatbuffers 模型位于 FLASH 的时候，默认情况下会复制到 PSRAM 或者 internal RAM 以提高推理性能。
variable	内存管理模块申请的内存，模型输入/输出以及中间的计算结果都会使用这部分空间。
others	类成员变量所需要的空间，heap_caps_aligned_alloc / heap_caps_aligned_calloc 申请过程中对齐的额外部分（很小）。

显示的内存类型: 每个类别的内部 RAM、PSRAM 和 FLASH 使用情况。

3.2.7 分析模型推理延迟

`profile_module()` 方法打印模型中每个模块（层）的详细延迟信息。

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

参数: - `sort_module_by_latency`：如果为 `true`，模块按延迟排序（最高优先）。如果为 `false`（默认），模块按拓扑顺序显示。

用法:

```
// 默认：拓扑顺序
model->profile_module();

// 按延迟排序（最高优先）
model->profile_module(true);
```

输出包括: - 模块名称 - 模块类型（操作类型）- 推理延迟（微秒，如果启用 `DL_LOG_LATENCY_UNIT` 则为周期数）- 末尾的总推理延迟

相关 API:

- `std::map<std::string, module_info> get_module_info()` - 以编程方式返回模块信息
- `void print_module_info(const std::map<std::string, module_info> &info, bool sort_module_by_latency = false)` - 从映射打印模块信息

3.2.8 组合性能分析：profile() 方法

`profile()` 方法结合了 `profile_memory()` 和 `profile_module()`，进行综合分析。

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

用法：

```
// 拓扑顺序的综合性能分析
model->profile();

// 按延迟排序的综合性能分析
model->profile(true);
```

这是获取内存和性能分析的最便捷方式。

3.3 如何进行模型推理

在本教程中，我们将介绍最基本的模型推理流程。参考例程

3.3.1 准备工作

安装 ESP_IDF

3.3.2 加载模型

如何加载模型

3.3.3 获取模型输入/输出。

```
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;
```

可以通过 `get_inputs()` 和 `get_outputs()` api 获得输入/输出的名字和对应的 `dl::TensorBase`。更多信息，请参阅*dl::TensorBase 文档* 。

备注： ESP-DL 的内存管理器会为每个模型的输入/中间结果/输出分配一整块的内存。由于它们共用这部分内存，所以当模型进行推理的时候，后面的结果会覆盖前面的结果。也就是说，`model_input` 中的数据，在执行完模型推理之后，可能就会被 `model_output` 或者其他中间结果所覆盖。

3.3.4 量化输入

8bit 和 16bit 量化的模型，分别接受 `int8_t` 和 `int16_t` 类型的输入。`float` 类型的输入必须先根据 `exponent` 量化成对应的整数类型之后才能喂入模型。计算公式：

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

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

其中：

- R 是要量化的浮点数。
- Q 是量化后的整数值，需要在 [MIN, MAX] 范围内进行裁剪。
- MIN 整数最小值，8bit 时，MIN = -128, 16bit 时，MIN = -32768。
- MAX 整数最大值，8bit 时，MAX = 127, 16bit 时，MAX = 32767。

量化单个值

```
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));
```

量化 dl::TensorBase

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

3.3.5 反量化输出

8bit 和 16bit 量化的模型，分别得到 `int8_t` 和 `int16_t` 类型的输出。必须根据 `exponent` 反量化之后才能得到浮点输出。计算公式：

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

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

其中：

- R’ 是反量化后恢复的近似浮点值。
- Q 是量化后的整数值。

反量化单个值

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

反量化 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 模型推理

请参阅：

- 参考例程
- *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 如何创建新模块（算子）

本教程将指导您在 `dl::module` 命名空间中创建一个新模块。`Module` 类是所有模块的基类，您将扩展这个基类来创建您的自定义模块。

备注： ESP-DL 中的模块接口应与 ONNX 对齐。

3.4.1 理解基类 `Module`

基类提供了几个必须在派生类中重写的虚方法。

- **方法：**
  - *dl::module::Module::Module()*：构造函数，用于初始化模块。
  - *dl::module::Module::~Module()*：析构函数，用于释放资源。
  - *dl::module::Module::get_output_shape()*：根据输入形状计算输出形状。
  - *dl::module::Module::forward()*：运行模块，高级接口。
  - *dl::module::Module::forward_args()*：运行模块，低级接口。
  - *dl::module::Module::deserialize()*：从序列化信息创建模块实例。
  - *dl::module::Module::print()*：打印模块信息。

更多信息，请参考 Module Class Reference。

3.4.2 创建新模块类

要创建一个新模块，您需要从 `Module` 基类派生一个新类并重写必要的方法。

示例：创建 `MyCustomModule` 类

更多示例，请参考 esp-dl/dl/module。

```
#include "module.h" // 包含定义 Module 类的头文件

namespace dl {
namespace module {
```

(下页继续)

(续上页)

```
class MyCustomModule : public Module {
public:
    // 构造函数
    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) {}

    // 析构函数
    virtual ~MyCustomModule() {}

    // 重写 get_output_shape 方法
    std::vector<std::vector<int>> get_output_shape(std::vector<std::vector<int>> &
↪input_shapes) override {
        // 实现根据输入形状计算输出形状的逻辑
        std::vector<std::vector<int>> output_shapes;
        // 示例：假设输出形状与输入形状相同
        output_shapes.push_back(input_shapes[0]);
        return output_shapes;
    }

    // 重写 forward 方法
    void forward(std::vector<dl::TensorBase *> &tensors, runtime_mode_t mode =␣
↪RUNTIME_MODE_AUTO) override {
        // 实现运行模块的逻辑
        // 示例：对张量执行某些操作
        for (auto &tensor : tensors) {
            // 对每个张量执行某些操作
        }
    }

    // 重写 forward_args 方法
    void forward_args(void *args) override {
        // 实现低级接口的逻辑
        // 示例：根据参数执行某些操作
    }

    // 从序列化信息反序列化模块实例
    static Module *deserialize(fbs::FbsModel *fbs_model, std::string node_name){
        // 实现反序列化模块实例的逻辑
        // 接口应与 ONNX 对齐
    }

    // 重写 print 方法
    void print() override {
        // 打印模块信息
        ESP_LOGI("MyCustomModule", "Module Name: %s, Quant type: %d", name.c_str(),
↪ quant_type);
    }
};

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

注册 MyCustomModule 类

当您实现了 MyCustomModule 类后，请在 dl_module_creator 中注册您的模块，使其全局可用。

```
void register_dl_modules()
{
```

(下页继续)

(续上页)

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

3.5 如何部署 MobileNetV2

在本教程中，我们介绍如何使用 ESP-PPQ 对预训练的 MobileNetV2 模型进行量化，并使用 ESP-DL 部署量化后的 MobileNetV2 模型。

3.5.1 准备工作

1. 安装 *ESP_IDF*
2. 安装 *ESP_PPQ*

3.5.2 模型量化

量化脚本

预训练模型

从 torchvision 加载 MobileNet_v2 的预训练模型，你也可以从 ONNX models 或 TensorFlow models 下载：

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

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

校准数据集

校准数据集需要和你的模型输入格式一致，校准数据集需要尽可能覆盖你的模型输入的所有可能情况，以便更好地量化模型。这里以 ImageNet 数据集为例，演示如何准备校准数据集。

使用 torchvision 加载 ImageNet 数据集：

```
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 默认配置量化

量化设置

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

量化结果

```
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%
/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%
```

(下页继续)

(续上页)

```
/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%
/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%
```

(下页继续)

(续上页)

```
/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*
```

量化误差分析

量化后的 top1 准确率只有 60.5%，和 float 模型的准确率 (71.878%) 相差较远，量化模型精度损失较大，其中：

- **累计误差 (Graphwise Error)**
  该模型的最后一层为 /classifier/classifier.1/Gemm，该层的累计误差为 25.591%。经验来说最后一层的累计误差小于 10%，量化模型的精度损失较小。
- **逐层误差 (Layerwise error)**
  观察 Layerwise error，发现大部分层的误差都在 1% 以下，说明大部分层的量化误差较小，只有少数几层误差较大，我们可以选择将误差较大的层使用 int16 进行量化。具体请看混合精度量化。

混合精度量化

量化设置

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

# 以下层使用int16进行量化
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))
```

量化结果

```
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%
/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%
```

(下页继续)

(续上页)

```
/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%
/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*
```

量化误差分析

将之前误差最大的层替换为 16 位量化后，可以观察到模型准确度明显提升，量化后的 top1 准确率为 69.550%，和 float 模型的准确率 (71.878%) 比较接近。该模型的最后一层 `/classifier/classifier.1/Gemm` 的累计误差为 9.117%。

层间均衡量化

该方法在论文 Data-Free Quantization Through Weight Equalization and Bias Correction 中提出。使用此方法时，需要将 MobilenetV2 模型中原来的 ReLU6 替换为 ReLU。

量化设置

```
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

# 将ReLU6 替换为 ReLU
model = convert_relu6_to_relu(model)
# 使用层间均衡
quant_setting = QuantizationSettingFactory.espdl_setting()
quant_setting.equalization = True
quant_setting.equalization_setting.iterations = 4
quant_setting.equalization_setting.value_threshold = .4
quant_setting.equalization_setting.opt_level = 2
quant_setting.equalization_setting.interested_layers = None
```

量化结果

```
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%
```

(下页继续)

(续上页)

```
/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%
/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%
```

(下页继续)

(续上页)

```
/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%

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

量化误差分析

注意到对 8bit 量化应用层间均衡有助于降低量化损失。模型最后一层 `/classifier/classifier.1/Gemm` 的累积误差为 8.965%。量化后的 top1 准确率为 69.800%，和 float 模型的准确率 (71.878%) 更加接近，比混合精度量化的量化精度更高。

备注： 如果想进一步降低量化误差，可以尝试使用 QAT (Auantization Aware Training)。具体方法请参考 PPQ QAT example。

3.5.3 模型部署

参考示例

图像分类基类

- dl_cls_base.hpp
- dl_cls_base.cpp

前处理

`ImagePreprocessor` 类中封装了常用的图像前处理流程，包括 `color conversion`, `crop`, `resize`, `normalization`, `quantize`。

- dl_image_preprocessor.hpp
- dl_image_preprocessor.cpp

后处理

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

3.6 如何部署 YOLO11n

在本教程中，我们介绍如何使用 ESP-PPQ 对预训练的 YOLO11n 模型进行量化，并使用 ESP-DL 部署量化后的 YOLO11n 模型。

3.6.1 准备工作

1. 安装 *ESP_IDF*
2. 安装 *ESP_PPQ*

3.6.2 模型量化

预训练模型

你可以从 Ultralytics release 下载预训练的 yolo11n 模型。

目前 ESP-PPQ 支持 ONNX、PyTorch、TensorFlow 模型。在量化过程中，PyTorch 和 TensorFlow 会先转化为 ONNX 模型，因此将与训练的 yolo11n 转化成 ONNX 模型。

具体来说，参考脚本：export_onnx.py 将预训练的 yolo11n 模型转换为 ONNX 模型。

在该脚本中，我们重载了 Detect 类的 forward 方法，具有以下优势：

- 更快的推理速度。与原始的 yolo11n 模型相比，将推理过程中 Detect 里与解码边界框相关的操作移至后处理中完成，从而显著减少了推理延迟。一方面，`Conv`，`Transpose`，`Slice`，`Split` 和 `Concat` 操作在推理过程中运行是非常耗时的。另一方面，在后处理阶段，模型推理的输出首先进行置信度筛选，然后再解码边界框，这大大减少了计算量，从而加快了整体推理速度。
- 更低的量化误差。ESP-PPQ 中的 `Concat` 和 `Add` 操作采用了联合量化。为了减少量化误差，由于 box 和 score 的范围差异较大，它们通过不同的分支输出，而不是拼接在一起。类似地，由于 `Add` 和 `Sub` 的输入的范围差异较大，相关计算被移到了后处理中进行，避免被量化。

校准数据集

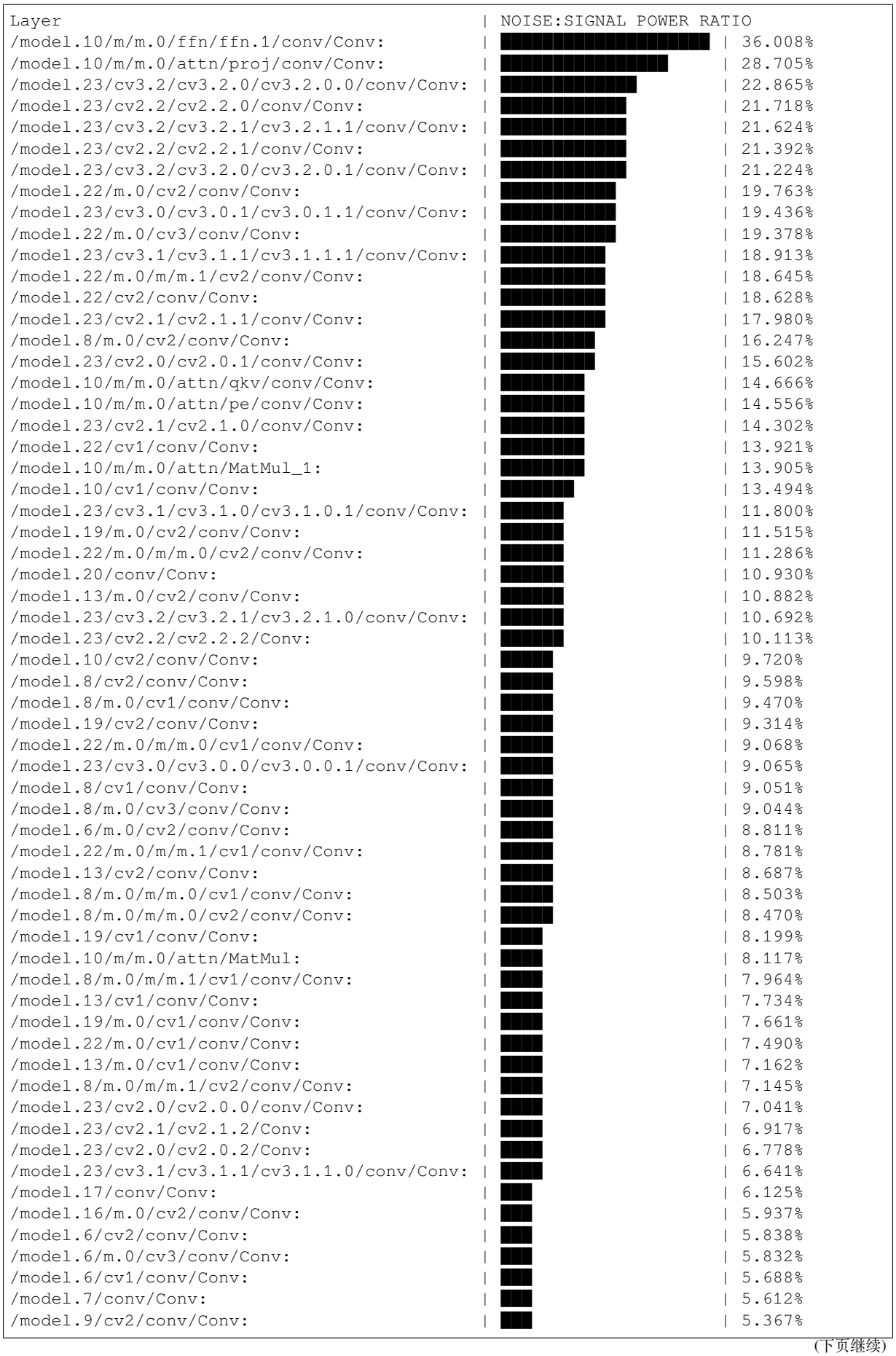
校准数据集需要和模型输入格式一致，同时尽可能覆盖模型输入的所有可能情况，以便更好地量化模型。本示例中，我们使用的校准集为 calib_yolo11n 。

8bit 默认配置量化

量化设置

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

量化结果


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%
/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%

(下页继续)

(续上页)

```
/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
/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%
```

(下页继续)

(续上页)

```
/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%
/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%
```

量化误差分析

在相同输入下，量化后的模型在 COCO val2017 上的 mAP50:95 仅为 30.7%，低于浮点模型，存在一定的精度损失：

- **累计误差 (Graphwise Error)**
  模型的输出层是 /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 和 /model.23/cv2.0/cv2.0.2/Conv，累计误差分别为 1.155%，10.113%，0.493%，6.917%，0.282% 和 6.778% 。通常，如果输出层的累计误差小于 10%，则量化模型的精度损失较小。
- **逐层误差 (Layerwise error)**
  观察逐层误差发现，所有层的误差均低于 1%，这表明所有层的量化误差都很小。

我们注意到，虽然所有层的逐层误差都很小，但是一些层的累计误差却较大。这可能与 yolo11n 模型中复杂的 CSP 结构有关，模型中 `Concat` 或 `Add` 层的输入可能具有不同的分布或尺度。我们可以选择使用 int16 对某些层进行量化，并采用算子分裂过程优化量化效果。有关详细信息，请参阅混合精度 + 算子分裂过程量化测试。

混合精度 + 算子分裂量化

量化设置

```
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']
```

量化结果

```
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%
```

(下页继续)

(续上页)

```
/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%
/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%
```

(下页继续)

(续上页)

```
/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%
/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%
```

(下页继续)

(续上页)

```
/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%
/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%
```

量化误差分析

在对逐层误差较高的层使用 16-bit 量化，并采用算子分裂过程后，在相同输入下，量化后的模型在 COCO val2017 上的 mAP50:95 提升至 33.4%；同时可以观察到输出层的累计误差明显减少。

模型的输出层/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 和/model.23/cv2.0/cv2.0.2/Conv 的累计误差分别为 0.771%，8.156%，0.339%，4.222%，0.190% 和 4.232%。

量化感知训练

为了进一步提高量化模型的精度，可以采用量化感知训练。本示例基于 8-bit 量化方式进行量化感知训练。

量化设置

- yolo11n_qat.py
- trainer.py

量化结果

```
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%
/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%
```

(下页继续)

(续上页)

```
/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%
/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%
```

(下页继续)

(续上页)

```
/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%
/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%
```

(下页继续)

(续上页)

```
/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%
```

量化误差分析

在对 8-bit 量化应用量化感知训练后，在相同输入下，量化后的模型在 COCO val2017 上的 mAP50:95 提升至 36.0%；同时输出层的累计误差大幅减少。相比前两种量化方式，量化感知训练后的 8-bit 量化模型可以在最快的推理速度下达到最高的量化精度。

模型的输出层/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 和/model.23/cv2.0/cv2.0.2/Conv 的累计误差分别为 0.746%，5.999%，0.480%，4.543%，0.386% 和 4.001%。

备注： 如果想要更快的模型推理速度，并且可以接受一定程度的精度损失，可以考虑在量化 YOLO11N 的时候将输入大小设置为 320x320。不同分辨率下的模型推理速度可以在 README.md 中找到。

3.6.3 模型部署

参考示例

目标检测基类

- dl_detect_base.hpp
- dl_detect_base.cpp

前处理

`ImagePreprocessor` 类中封装了常用的图像前处理流程，包括 `color conversion`, `crop`, `resize`, `normalization`, `quantize`。

- dl_image_preprocessor.hpp
- dl_image_preprocessor.cpp

后处理

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

3.7 如何部署 YOLO11n-pose

在本教程中，我们介绍如何使用 ESP-PPQ 对预训练的 YOLO11n-pose 模型进行量化，并使用 ESP-DL 部署量化后的 YOLO11n-pose 模型。

3.7.1 准备工作

1. 安装 *ESP_IDF*
2. 安装 *ESP_PPQ*

3.7.2 模型量化

预训练模型

你可以从 Ultralytics release 下载预训练的 yolo11n-pose 模型。

目前 ESP-PPQ 支持 ONNX、PyTorch、TensorFlow 模型。在量化过程中，PyTorch 和 TensorFlow 会先转化为 ONNX 模型，因此将与训练的 yolo11n-pose 转化成 ONNX 模型。

具体来说，参考脚本：export_onnx.py 将预训练的 yolo11n-pose 模型转换为 ONNX 模型。

在该脚本中，我们重载了 Pose 类的 forward 方法，具有以下优势：

- 更快的推理速度。与原始的 yolo11n-pose 模型相比，将推理过程中 Pose 里与解码边界框相关的操作移至后处理中完成，从而显著减少了推理延迟。一方面，`Conv`，`Transpose`，`Slice`，`Split` 和 `Concat` 操作在推理过程中运行是非常耗时的。另一方面，在后处理阶段，模型推理的输出首先进行置信度筛选，然后再解码边界框，这大大减少了计算量，从而加快了整体推理速度。
- 更低的量化误差。ESP-PPQ 中的 `Concat` 和 `Add` 操作采用了联合量化。为了减少量化误差，由于 box 和 score 的范围差异较大，它们通过不同的分支输出，而不是拼接在一起。类似地，由于 `Add` 和 `Sub` 的输入的范围差异较大，相关计算被移到了后处理中进行，避免被量化。

校准数据集

校准数据集需要和模型输入格式一致，同时尽可能覆盖模型输入的所有可能情况，以便更好地量化模型。本示例中，我们使用的校准集为 calib_yolo11n-pose 。

8bit 默认配置量化

量化设置

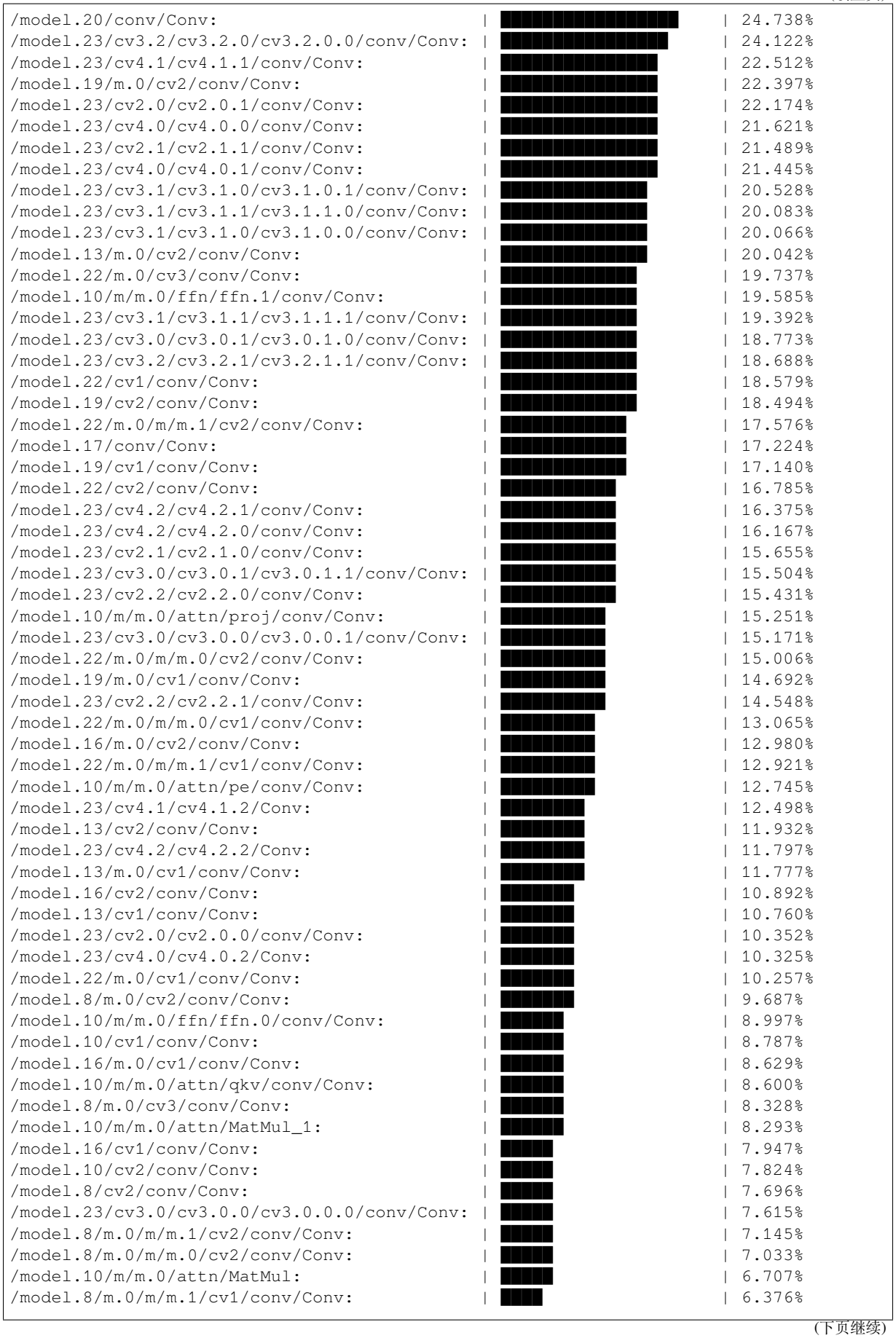
```
target="esp32p4"
num_of_bits=8
batch_size=32
quant_setting = QuantizationSettingFactory.espdl_setting() # default setting
```

量化结果

```
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%
```

(下页继续)

(续上页)


/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%
/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%

(下页继续)

(续上页)

```
/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%
/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%
```

(下页继续)

(续上页)

```
/model.23/cv2.1/cv2.1.1/conv/Conv:           |                       | 0.018%
/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%
/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%
```

(下页继续)

(续上页)

```
/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%
```

量化误差分析

在相同输入下，量化后的模型在 COCO 上的 Pose mAP50:95 仅为 43.1%，低于浮点模型（50.0%），存在一定的精度损失。

量化感知训练

为了进一步提高量化模型的精度，可以采用量化感知训练。本示例基于 8-bit 量化方式进行量化感知训练。

量化设置

- yolo11n-pose_qat.py
- trainer.py

量化结果

```
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%
/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%
```

(下页继续)

(续上页)

```
/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%
/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%
```

(下页继续)

(续上页)

```
/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%
/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%
```

(下页继续)

(续上页)

```
/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%
/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%
```

量化误差分析

在对 8-bit 量化应用量化感知训练后，在相同输入下，量化后的模型在 COCO 上的 Pose mAP50:95 提升至 44.9%；同时输出层的累计误差大幅减少。相比 8-bit 后量化方式，量化感知训练后的 8-bit 量化模型可以在相同的推理速度下达到最高的量化精度。

3.7.3 模型部署

example

目标检测基类

- dl_detect_base.hpp
- dl_detect_base.cpp

前处理

`ImagePreprocessor` 类中封装了常用的图像前处理流程，包括 `color conversion`, `crop`, `resize`, `normalization`, `quantize`。

- dl_image_preprocessor.hpp
- dl_image_preprocessor.cpp

后处理

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

3.8 如何部署流式模型

时间序列模型如今被应用在许多领域，例如，音频领域。而音频模型在部署时通常有两种模式：

- Offline 模式：模型需要一次性接收完整的音频数据（例如整个语音文件），然后进行整体处理。
- Streaming 模式：流式模式下，模型逐帧（逐块）接收音频数据，实时处理并输出中间结果。

在本教程中，我们来介绍如何使用 ESP-PPQ 量化流式模型，并使用 ESP-DL 部署量化后的流式模型。

3.8.1 准备工作

1. 安装 *ESP_IDF*
2. 安装 *ESP_PPQ*

3.8.2 模型量化

参考示例

如何转换为流式模型

时间序列模型种类繁多，这里仅以 Temporal Convolutional Network(TCN) 为例，不熟悉的可自行查找资料了解，这里不过多介绍其细节。其它模型需根据自身情况，量体裁衣。

该示例代码中构建了一个 TCN 模型：models.py (模型非完整，仅用于演示)。

ESP-PPQ 提供了自动流式转换功能，可以简化创建流式模型的过程。通过 `auto_streaming=True` 参数，ESP-PPQ 自动处理流式推理所需的模型转换。

备注:

- Offline 模式，模型输入是一段完整数据，input shape 在时间维度上的 size 一般比较大（例如 `[1, 16, 15]`）。
- Streaming 模式，模型输入是连续数据，在时间维度上的 size 较小，匹配实时处理的块大小（例如 `[1, 16, 3]`）。

自动流式转换

ESP-PPQ 通过量化过程中的 `auto_streaming=True` 参数提供自动流式转换功能。启用此标志后，ESP-PPQ 会自动转换模型以支持流式推理：

1. 分析模型结构以识别适当的分块点
2. 创建内部状态管理以在块之间保持上下文
3. 生成适合流式场景的优化代码

以下是如何使用自动流式功能的示例：

```
# 导出非流式模型
quant_ppq_graph = espdl_quantize_torch(
    model=model,
    espdl_export_file=ESPDL_MODEL_PATH,
    calib_dataloader=dataloader,
    calib_steps=32,  # 校准步数
    input_shape=INPUT_SHAPE,  # 离线模式的输入形状
    inputs=None,
    target=TARGET,  # 量化目标类型
    num_of_bits=NUM_OF_BITS,  # 量化位数
    dispatching_override=None,
    device=DEVICE,
    error_report=True,
    skip_export=False,
    export_test_values=True,
    verbose=1,  # 输出详细日志信息
)

# 使用自动转换导出流式模型
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,  # 启用自动流式转换
    streaming_input_shape=[1, 16, 3],  # 流式模式的输入形状
    streaming_table=None,
)
```

3.8.3 模型部署

参考示例，该示例使用预生成的数据来模拟实时数据流。

备注： 基础的模型加载和推理方法，可参考其它文档，这里不再赘述：

- 如何加载和测试模型
- 如何进行模型推理

在流式模式下，模型按时间接收数据块，而不是要求一次性获得整个输入。流式模型依次处理这些块，同时在块之间保持内部状态。部署代码负责将输入分解为适当的块并将其馈送到模型。见 app_main.cpp 如下代码块：

```
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.");
        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;
}
```

这种方法允许模型通过将长序列分解为更小、更易管理的块来高效处理。每个块依次馈送到模型中，内部状态自动维护以确保跨块的连续性。

备注：

- 块的数量是根据完整输入大小与流式模型输入大小的比率计算的。
- ESP-DL 流式模型自动处理内部状态管理，使部署变得简单。
- 流式模型的输出应与等效离线模型输出的最后部分匹配。

Chapter 4

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 **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.

参数

- **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

inline virtual **~TensorBase**()

Destroy the *TensorBase* object.

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

Assign tensor to this tensor.

参数 **tensor** –
返回 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.

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

返回 true if assign successfully, otherwise false.

inline int **get_size**()

Get the size of Tensor.

返回 the size of Tensor.

inline int **get_aligned_size**()

Get the aligned size of Tensor.

返回 the aligned size of Tensor.

inline size_t **get_dtype_bytes**()

Get the dtype size, in bytes.

返回 the size of dtype.

inline const char ***get_dtype_string**()

Get the dtype string of Tensor.

返回 the string of Tensor’s dtype.

inline int **get_bytes**()

Get the bytes of Tensor.

返回 the bytes of Tensor.

inline int **get_aligned_bytes**()

Get the bytes of Tensor.

返回 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.

返回 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.

返回 the pointer of Tensor’s data

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

Set the data pointer of Tensor.

参数 **data** –point to data memory
返回 *TensorBase*& self

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

Get the shape of Tensor.

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

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

Set the shape of Tensor.

参数 **shape** –the shape of Tensor.
返回 Tensor.

inline int **get_exponent**()

Get the exponent of Tensor.

返回 int the exponent of Tensor

inline dtype_t **get_dtype**()

Get the data type of Tensor.

返回 dtype_t the data type of Tensor

inline uint32_t **get_caps**()

Get the memory flags of Tensor.

返回 uint32_t the memory flags of Tensor

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

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

参数 **shape** –the target shape
返回 *TensorBase* *self

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

Flip the input Tensor along the specified axes.

参数 **axes** –the specified axes
返回 *TensorBase*& self

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

Reverse or permute the axes of the input Tensor.

参数

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

返回 *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.

参数

- **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.

返回 *TensorBase* *self

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

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

参数 **tensor** –Input tensor pointer

返回
- true: same shape
- false: not

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

Compare the shape and data of two Tensor.

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

返回 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.

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

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

返回 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.

参数
- **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

返回 Output tensor pointer

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

Pad input tensor.

参数
- **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

返回 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.

参数
- **gt_elements** –The ground truth elements
- **epsilon** –The max error of two element

- **verbose** –If true, print the detail of results

返回 true if all elements are equal otherwise false

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

Get the index of element.

参数 **axis_index** –The coordinates of element
返回 int the index of element

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

Get the coordinates of element.

参数 **index** –The index of element
返回 The coordinates of element

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

Get a element of Tensor by index.

参数 **index** –The index of element
返回 The element of tensor

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

Get a element of Tensor.

参数 **axis_index** –The index of element
返回 The element of tensor

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

Set preload address of Tensor.

参数

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

返回 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.

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

参数

- **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.".

参数

- **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*

参数 **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

int **exponent**

exponent of element

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.

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

参数

- **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.

参数

- **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.

返回 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.

参数 **input_shapes** –Input shapes
返回 outputs shapes

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

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

参数

- **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.

参数 **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

参数

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

inline virtual void **preload**()

Perform a preload operation.

> **警告:** 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.

参数

- **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.

参数

- **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

参数

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

返回 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.

参数

- **`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.

参数

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

返回 *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.

返回 *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.

参数

- **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.

参数

- **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.

参数

- **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.

参数

- **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.

参数

- **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.

返回

- 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.

参数

- **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.

返回

- 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.

参数

- **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.

返回

- ESP_OK Success
- ESP_FAIL Failed

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

Load model graph and parameters from Flatbuffers model.

参数 **fbs_model** –The FlatBuffers model

返回

- 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.

参数

- **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.

参数 **mode** –Runtime mode.

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

Run the model module by module.

参数

- **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.

参数

- **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.

返回 esp_err_t

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

Get memory info.

返回 Memory usage statistics on internal and PSRAM.

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

Get module info.

返回 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.

参数

- **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)

参数 **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.

参数 **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.

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

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

Get the only input of model.

返回 TensorBase*

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

Get input of model by name.

参数 **name** –input name
返回 TensorBase*

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

Get intermediate *TensorBase* of model.

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

---

参数 **name** –The name of intermediate Tensor. operators.
返回 The intermediate TensorBase*.

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

Get outputs of model.

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

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

Get the only output of model.

返回 TensorBase*

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

Get output of model by name.

参数 **name** –output name
返回 TensorBase*

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

Get the model' s metadata prop.

参数 **key** –The key of metadata prop
返回 The value of metadata prop

virtual void ***print***()

Print the model.

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

Get the fbs model instance.

返回 *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.

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

返回 int Returns the index of the added tensor.

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

Push back a tensor.

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

返回 int Returns the index of the added tensor.

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

Updates the tensor at the specified index.

参数
- **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.

参数 **index** –The index of the tensor.

返回 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.

参数 **name** –The name of the tensor.

返回 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.

参数 **name** –The name of the tensor.

返回 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.

参数 **name** –The name of the tensor.

返回 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.

返回 int Returns the number of variable tensors.

inline int **get_parameter_count**()

Gets the count of parameter tensors.

返回 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.

参数

- **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.

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

inline void ***get_psram_root**()

Gets the pointer to the PSRAM root.

返回 Void* Returns the pointer to the PSRAM root.

inline void ***get_internal_root**()

Gets the pointer to the internal root.

返回 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.

参数

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

返回 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.

参数 **mem_info** –The size of the memory used by the variables in bytes.

返回 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 **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.

参数 **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.

参数

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

返回 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.

参数

- **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.

参数 **tensor** –Inplace leader tensor

inline void **set_inplace_follower_tensor**(*TensorInfo* *tensor)

Set the inplace follower tensor object.

参数 **tensor** –Inplace follower tensor

inline *TensorInfo* ***get_inplace_follower_tensor**()

Get the inplace follower tensor object.

返回 TensorInfo* Inplace follower tensor

void **update_time**(int new_time)

Update Tensor lifetime.

参数 **new_time** –new tensor lifetime

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

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

参数

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

返回 TensorBase*

inline bool **is_inplaced**()

Is inplaced or not.

返回 true if inplaced else false

inline uint32_t **get_offset**()

Get the tensor offset.

返回 uint32_t

inline void **set_offset**(uint32_t offset)

Set the tensor offset.

参数 **offset** –

inline uint32_t **get_internal_offset**()

Get the internal offset.

返回 uint32_t

inline bool **get_internal_state**()

Get the internal state.

返回 true if is internal else false

inline void **set_internal_state**(bool is_internal)

Set the internal state.

参数 **is_internal** –

inline void **set_internal_offset**(uint32_t offset)

Set the internal offset.

参数 **offset** –

inline int **get_time_end**()

Get the liftetime end.

返回 int

inline int **get_time_begin**()

Get the liftetime begin.

返回 int

inline size_t **get_size**()

Get the tensor size.

返回 size_t

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

Get the tensor name.

返回 std::string

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

Get the tensor shape.

返回 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.

参数

- **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.

参数

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

inline **~MemoryChunk**()

Destroy the Memory Chunk object.

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

Merge continuous free chunk.

参数 **chunk** –
返回 MemoryChunk*

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

Insert tensor into free chunk.

参数 **tensor** –
返回 MemoryChunk*

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

Extend free chunk and insert tensor.

参数 **tensor** –
返回 MemoryChunk*

inline void **free**()

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

size_t **get_aligned_size**(size_t size)

get aligned size, which is 16/alignment bytes aligned

参数 **size** –
返回 size_t

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.

参数

- **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.

参数

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

返回 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.

参数

- **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.

参数

- **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.

返回 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.

参数

- **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.

返回 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.

参数

- **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.

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

int **get_model_num**()

Get the number of models.

返回 The number of models

void **list_models**()

List all model’ s name.

const char ***get_model_location_string**()

Get the model location string.

返回 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.

参数

- **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.

返回 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.

参数

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

返回 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.

参数

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

返回 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.

参数

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

- **ret_value** –The attribute value.

返回 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.

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

返回 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.

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

返回 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.

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

返回 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.

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

返回 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.

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

返回 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.

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

返回 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.

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

返回 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.

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

返回 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.

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

返回 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.

参数 **node_name** –The name of operation

返回 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.

参数
- **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

返回 *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.

参数
- **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

返回 *dl::TensorBase**

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

return true if the variable is a parameter

参数 **name** –Variable name

返回 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.

参数 **tensor_name** –The name of Tensor.
返回 uint8_t * The pointer of raw data.

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

Get the element type of tensor tensor.

参数 **tensor_name** –The tensor name.
返回 FlatBuffers::Dl::TensorDataType

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

Get the shape of tensor.

参数 **tensor_name** –The name of tensor.
返回 std::vector<int> The shape of tensor.

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

Get the exponents of tensor.

> **警告:** 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.

参数 **tensor_name** –The name of tensor.
返回 The exponents of tensor.

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

Get the element type of value_info.

参数 **var_name** –The value_info name.
返回 dl::dtype_t

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

Get the shape of value_info.

参数 **var_name** –The value_info name.
返回 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.

参数 **var_name** –The value_info name.
返回 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.

参数 **tensor_name** –The name of test input tensor.
返回 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.

参数 **tensor_name** –The name of test output tensor.
返回 uint8_t * The pointer of raw data.

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

Get the test input tensor.

参数 **tensor_name** –The name of test input tensor.

返回 The pointer of tensor.

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

Get the test output tensor.

参数 **tensor_name** –The name of test output tensor.
返回 The pointer of tensor.

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

Get the name of test outputs.

返回 the name of test outputs

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

Get the graph inputs.

返回 the name of inputs

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

Get the graph outputs.

返回 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.

返回 the name of model

int64_t **get_model_version**()

Get the model version.

返回 The version of model

std::string **get_model_doc_string**()

Get the model doc string.

返回 The doc string of model

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

Get the model' s metadata prop.

参数 **key** –The key of metadata prop
返回 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.

参数

- **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.

索引

C

D

F