**OpenHarmony竞赛训练营**

**赛题：题目九：鸿蒙AI推理网络的精度自动对比工具**

|  |  |
| --- | --- |
| 学校名称： | 华中科技大学 |
| 团队名称： | 加冰陈化 |
|  |  |
| 队长： | 刘浩毅 |

OpenHarmony竞赛训练营组委会

2024年9月

目 录

[1 简介 3](#_Toc179365147)

[1.1 背景 3](#_Toc179365148)

[1.2 目的 3](#_Toc179365149)

[2 设计描述 4](#_Toc179365150)

[2.1 总体设计 4](#_Toc179365151)

[2.2 实现思路 5](#_Toc179365152)

[2.3 系统结构 5](#_Toc179365153)

[2.3.1 模块划分 5](#_Toc179365154)

[2.3.2 系统架构说明 7](#_Toc179365155)

[2.3.3 文件结构 8](#_Toc179365156)

[2.4 模块功能描述 9](#_Toc179365157)

[2.4.1 模块1功能描述 Import & Config 9](#_Toc179365158)

[2.4.2 模块/子系统2功能描述 10](#_Toc179365159)

[2.4.3 模块/子系统3功能描述 11](#_Toc179365160)

[2.4.4 模块/子系统4功能描述 11](#_Toc179365161)

[2.4.5 模块/子系统5功能描述 12](#_Toc179365162)

[2.5 业务/实现流程说明 13](#_Toc179365163)

[2.5.1 推理结果对比处理流程 13](#_Toc179365164)

[2.5.2 自定义映射处理流程 14](#_Toc179365165)

[2.6 接口描述 14](#_Toc179365166)

[2.6.1 调用接口 14](#_Toc179365167)

[2.6.2 提供接口 15](#_Toc179365168)

[2.7 UI设计 16](#_Toc179365169)

[3 其他 17](#_Toc179365170)

[3.1 成员分工 17](#_Toc179365171)

[3.2 困难与思考 17](#_Toc179365172)

[3.3 参考 18](#_Toc179365173)

# 简介

## 背景

ONNX推理网络基于鸿蒙AI子系统(MindSpore Lite)部署在手机等终端上，需做严格的推理精度验证，保证推理结果满足推理网络的目标精度要求。在交付完成经常会碰到推理结果偏离了目标精度，需要做精度误差引入的定位定界。

目前需要手动修改推理脚本获取网络推理的各层计算结果，然后依赖人工分析去对比ONNX和MindIR的输出结果，对比效率极低，影响推理业务上线速度。

使用Netron将推理模型进行可视化如下图，ONNX模型与MindSpore Lite的算子推理结果需要一一对应比较验证。

图表

描述已自动生成

## 目的

对于ONNX的模型在MindSpore Lite上推理的精度自动化对比能力主要问题和挑战主要包括以下两个方面：

1、原生ONNX-RT没有不修改推理脚本的逐层数据Dump能力；

2、ONNX和MindSpore Lite在IR设计上存在差异，映射关系复杂，精度自动对比难度大。

赛题的目的是实现一个可自动化的精度对比工具：输入一个类似llama网络（单机）暂不考虑大模型分布式场景，原始格式为onnx格式，通过Convert-Lite工具转为MindIR。原始格式运行在onnx runtime上，MindIR运行在MindSpore Lite并获取二者的运行数据。该工具对获取的数据进行对比，输出如下信息：

1、可自动对比的数据的比对结果；

2、对于不可对比的数据给出是哪些数据；

3、给出自动化对比比例。

基础目标如下：

常用网络(Resnet系列/Yolo系列/SSD/Bert/Transformer等)的运行数据自动化对比正确，且完成自动对比比列70%+；

提供无法自动对比的用户自定义注入对比能力。

挑战目标如下：

构建ONNX的脚本无修改数据Dump能力并提交ONNX社区；

常用网络(Resnet系列/Yolo系列/SSD/Bert/Transformer等)的运行数据自动化对比正确，且完成自动对比比列90%+。

为了确保ONNX模型在MindSpore Lite上的精度自动化对比能够顺利实现，文档将围绕以下几个具体目的展开：

实现逐层数据的对比能力。设计一种方案，使得在不修改ONNX推理脚本的前提下，能够对推理过程中的逐层数据进行Dump，从而便于精度对比分析。

探究ONNX和MindSpore Lite之间的IR映射关系。通过研究ONNX与MindSpore Lite中间表示（IR）之间的差异，制定清晰的映射机制，降低两者间的对比难度，为精度验证提供基础。

设计精度对比的自动化流程。构建一个自动化流程，能够批量化、系统化地完成ONNX模型与MindSpore Lite推理结果的精度对比，提升效率和准确性。

提升对比工具的可扩展性。确保所设计的工具或框架具有良好的扩展能力，能够适应不同模型、不同推理场景下的精度对比需求。

# 设计描述

## 总体设计

基于以上的目标，本项目设计了MindAcc来完成。

MindAcc是一个JupyterNotebook笔记本，使用Python完成大部分实现，大致可以分为三层：

UI层：使用Gradio设计UI界面，并通过Gradio的部署能力暴露端口使得用户可以通过浏览器进行访问，经过端口映射后也可以远程访问MindAcc。

模型层：通过实现三个模型提供相关API，组合使用即可实现模型加载、输入生成、推理与数据dump、映射生成、映射自定义、结果分析、生成统计图等功能，由UI层触发。

LIB层：使用MindSpore Lite运行库（linux elf与动态链接库）提供MindSpore Lite模型的推理、转换与数据dump能力，使用python的onnx、onnxdump、onnxruntime提供ONNX模型的推理与数据dump能力。使用numpy等库提供分析能力。

其关系如下图所示：

图示

描述已自动生成

## 实现思路

对于主要的三层划分，本项目根据问题所需实现了它们的功能：

UI层：用户可以通过上传文件、点击按钮等交互方式使用交互功能，最终生成对比结果并下载。应用界面使用了 Gradio 的 Blocks、HTML、Accordion、Row、Column、Group、Button、File、Textbox、DataFrame、CheckboxGroup 等组件来构建。

模型层：定义三个类MindaccModel、MindAccMapper、MindAccAnalyzer分别实现相应功能。MindaccModel提供加载模型、保存模型、获取模型信息等功能。MindAccMapper用于生成模型节点的映射关系。它可以解析模型结构，生成节点之间的映射关系图。MindAccAnalyzer用于对比和分析模型的推理结果，对比不同模型的输出，生成对比分析报告，帮助用户评估模型的性能和准确性。

LIB层：进行环境变量隔离与设置，构建一套LIB的API执行最终的命令执行等。

## 系统结构

### 模块划分

本系统根据需求分析，规划设计了以下主要功能模块，并使用Gradio进行可视化：

**Import & Config**：

该模块负责导入必要的库和配置环境。主要包括以下功能：导入 Python 标准库和第三方库，如 numpy, pandas, onnx, onnxruntime, gradio 等。配置环境变量，确保 MindSpore Lite 运行环境正确设置。初始化 Jupyter Notebook 环境，确保代码单元格可以顺利运行。

**模型管理模块：**

该模块负责将 ONNX 模型转换为 MindSpore Lite 模型。主要功能包括：读取 ONNX 模型文件。使用 MindSpore Lite 提供的工具将 ONNX 模型转换为 MindSpore Lite 模型格式。支持多种模型格式的转换，确保兼容性。该模块负责保存和加载模型的状态。主要功能包括：保存转换后的 MindSpore Lite 模型到指定路径。加载已保存的模型，方便后续推理和分析。

**映射器模块：**

该模块负责自定义算子映射。主要功能包括：定义 ONNX 模型与 MindSpore Lite 模型之间的算子映射规则。支持用户自定义算子映射，确保特殊算子可以正确转换。

**分析器模块：**

该模块负责对比分析推理结果。主要功能包括：运行 MindSpore Lite 和 ONNX 模型的推理，并获取推理结果。对比两种模型的推理结果，支持多种分析指标，如精度、速度等。提供可视化对比结果，方便用户直观了解模型性能差异。

**Gradio 可视化模块：**

该模块负责提供 Gradio 界面，方便用户使用 MindAcc 工具。主要功能包括：提供友好的用户界面，支持模型转换、推理运行、结果对比等功能。支持实时交互，用户可以通过界面输入参数，实时查看结果，提供映射规则的可视化界面，方便用户查看和修改映射规则。集成 Gradio，可在浏览器中打开，方便用户操作。

模块划分如下图所示：

图示

描述已自动生成

### 系统架构说明

MindAcc的主要处理流程如下。

1.用户打开 [main.ipynb](vscode-file://vscode-app/c:/Users/23629/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.esm.html) 文件，运行所有单元格。

2.用户 打开 Gradio 界面。

3.Gradio 界面初始化 MindAccModel等。

4.用户提交模型文件。

5.Gradio 界面 加载 ONNX 模型到 MindAccModel。

6.MindAccModel 使用 ONNX Runtime 运行 ONNX 模型推理。

7.MindAccModel 使用 MindSpore Lite 运行 MindSpore Lite 模型推理。

8.MindAccModel dump 推理数据。

9.MindAccModel 将推理数据传递给 MindAccMapper 进行映射。

10.MindAccMapper 将映射后的数据提供给 MindAccAnalyzer。

11.MindAccAnalyzer 对比推理结果。

12. MindAccAnalyzer 将对比结果返回给 Gradio 界面。

13.Gradio 界面显示对比结果给用户。

图示

低可信度描述已自动生成

### 文件结构

MindAcc的目录结构如下。

mindacc

├─ .gitignore

├─ LICENSE

├─ mindspore-lite-2.3.1-linux-x64   // MindSpore Lite 端侧运行环境

├─ input                            // 模型输入数据

├─ output                           // 模型输出数据

├─ model                            // 模型文件

├─ README.assets                    // README.md 图片等资源

├─ README.md                        // 项目说明文档

├─ environment.yml                  // conda 环境配置文件

├─ main.ipynb                       // 主程序

└─ requirements.txt                 // python 依赖

其中的main.ipynb的大纲如下所示。

main.ipynb

├─ Import & Config                      // 导入模块与配置

│  ├─ import            // 导入模块

│  ├─ logger            // 日志配置

│  └─ mslite\_config     // MindSpore Lite 环境配置

├─ Class MindAcc Model                  // 模型管理

│  ├─ load              // 加载模型

│  ├─ run\_ms\_converter  // 运行模型转换

│  ├─ input\_generate    // 生成随机输入

│  ├─ input\_load        // 加载输入数据

│  ├─ run\_ms\_dump       // 运行 MindSpore Lite 推理

│  └─ run\_onnx\_dump     // 运行 ONNX 推理

├─ Class MindAcc Mapper                 // 映射器

│  ├─ get\_ms\_bin\_info   // 获取 mslite 输出文件信息

│  ├─ read\_ms\_output    // 读取 mslite 输出文件夹

│  ├─ simple\_map        // 简单映射

│  └─ get\_map\_result    // 获取映射结果

├─ Class MindAcc Analyzer               // 分析器

│  ├─ cosine\_similarity // 余弦相似度

│  ├─ relative\_euclidean\_distance   // 相对欧氏距离

│  ├─ max\_absolute\_error    // 最大绝对误差

│  ├─ mean\_absolute\_error   // 平均绝对误差

│  ├─ root\_mean\_square\_error    // 均方根误差

│  ├─ max\_relative\_error    // 最大相对误差

│  ├─ mean\_relative\_error   // 平均相对误差

│  ├─ accumulated\_relative\_error    // 累积相对误差

│  ├─ standard\_deviation    // 标准差

│  └─ kullback\_leibler\_divergence   // KL散度

└─ Gradio Interface                     // Gradio 界面

   ├─ process\_model         // 函数绑定

   ├─ process\_input

   ├─ random\_input

   ├─ run\_infer

   ├─ run\_map

   ├─ run\_compare

   ├─ gr.Blocks             // Gradio 布局组件

   └─ launch                // Gradio 启动

## 模块功能描述

### 模块1功能描述 Import & Config

Import & Config模块描述如下。

导入库：导入了多个库，包括 collections, gradio, json, logging, numpy, os, onnx, onnxruntime, onnxdumper, pandas, pathlib, shutil, 和 re。

日志配置：使用 logging 库配置日志记录器，设置日志级别为 INFO。

配置 MindSpore Lite 环境变量：定义 MindSpore Lite 的根路径 PACKAGE\_ROOT\_PATH。设置转换器和基准测试工具的路径。配置 LD\_LIBRARY\_PATH 环境变量，以包含 MindSpore Lite 的运行时库和转换器库。

配置基准测试工具：定义基准测试工具的配置字典 mslite\_benchmark\_config，包括 dump 设置。

写入配置文件：将基准测试配置写入 dump\_config.json 文件。设置 MINDSPORE\_DUMP\_CONFIG 环境变量，指向生成的配置文件。

创建必要的目录：定义并创建 input, model, output 以及基准测试输出目录。记录当前 dump 目录结构。

### 模块/子系统2功能描述 MindAcc Model

MindAcc Model模块定义了一个名为 MindaccModel 的类，用于处理 ONNX 模型的加载、转换、输入生成和推理等操作。以下是该模块的主要功能和方法：

MindaccModel: 该类包含多个属性和方法，用于管理和操作 ONNX 模型。

属性：

path: 模型文件的路径。

name: 模型的名称。

onnx\_session: ONNX 推理会话。

input\_nodes: 模型的输入节点列表。

output\_nodes: 模型的输出节点列表。

precision: 模型输入的精度类型。

onnx\_input: ONNX 模型的输入数据。

ms\_output\_path: 转换后的模型输出路径。

onnx\_output\_path: ONNX 模型推理输出路径。

compare\_map: 比较映射，用于存储比较结果。

compare\_result: 比较结果的数据框。

方法：

load(model\_file: gr.File): 加载 ONNX 模型文件，初始化推理会话和节点信息。

run\_ms\_converter(optimize="") -> Path: 运行模型转换器，将 ONNX 模型转换为其他格式。

input\_generate(seed=0) -> dict: 生成随机输入数据，并保存为文件。

input\_load(input\_data: gr.File) -> None: 加载输入数据文件，并转换为 ONNX 输入格式。

run\_ms\_dump() -> Path: 运行基准测试，生成模型推理的转储数据。

run\_onnx\_dump(dump\_path="onnx\_dumpinrun.npz") -> dict: 运行 ONNX 模型推理，并保存输出数据。

该模块通过定义一系列方法，提供了从模型加载、输入生成到模型推理的完整流程，适用于需要对 ONNX 模型进行操作和分析的场景。

### 模块/子系统3功能描述MindAcc Mapper

MindAccMapper 类用于解析和映射 MindSpore 输出文件与 ONNX 模型输出之间的关系。以下是该类的主要功能和方法：

类方法：

get\_ms\_bin\_info(cls, ms\_bin\_file: Path) ：从文件名中解析出 shape 和 dtype 信息。通过使用正则表达式匹配文件名格式{op\_name}\_{input\_output\_index}\_{shape}\_{data\_type}\_{format}.bin。返回包含解析信息的字典。

read\_ms\_output(cls, ms\_output\_path: Path)：读取指定目录下的 .bin 文件。调用 get\_ms\_bin\_info 方法解析文件信息。返回包含解析结果的字典。

实例方法：

init(self, onnx\_model: str, onnx\_dump\_file: Path, ms\_dump\_dir: Path, extra\_rules: dict = {})：初始化方法，加载 ONNX 模型和 MindSpore 输出文件。解析 ONNX 模型和 MindSpore 输出文件。

simple\_map(self)：将 MindSpore 输出与 ONNX 模型输出进行简单映射。根据 shape 和 dtype 进行匹配。返回映射结果的字典。

get\_map\_result(self)：获取映射结果的统计信息。返回映射成功率、未映射列表和映射详情。

该模块通过定义一系列方法，提供了从模型加载、输入生成到模型推理的完整流程，适用于需要对 ONNX 模型进行操作和分析的场景。

### 模块/子系统4功能描述MindAcc Analyzer

MindAccAnalyzer 是一个用于分析模型精度的类。它包含多个分析器，每个分析器都是一个用于计算不同误差或相似度指标的 lambda 函数。该类包含以下分析器：

cosine\_similarity: 计算两个向量的余弦相似度。

relative\_euclidean\_distance: 计算两个向量的相对欧几里得距离。

max\_absolute\_error: 计算两个向量之间的最大绝对误差。

mean\_absolute\_error: 计算两个向量之间的平均绝对误差。

root\_mean\_square\_error: 计算两个向量之间的均方根误差。

max\_relative\_error: 计算两个向量之间的最大相对误差。

mean\_relative\_error: 计算两个向量之间的平均相对误差。

accumulated\_relative\_error: 计算两个向量之间的累积相对误差。

standard\_deviation: 计算两个向量之间误差的标准差。

kullback\_leibler\_divergence: 计算两个向量之间的 Kullback-Leibler 散度。

### 模块/子系统5功能描述Gradio Interface

Gradio Interface模块使用 Gradio 库创建了一个名为 MindAcc 的用户界面，用于对比 MindSpore Lite 和 ONNX 模型的推理结果。以下是该界面的主要功能和布局：

标题和描述：使用 gr.HTML 显示标题 "MindAcc" 及其描述，说明该工具的用途和功能。

使用说明：使用 gr.Accordion 展示使用说明，详细描述了如何使用该工具的每个步骤。

输入和选项：使用 gr.Row 和 gr.Column 布局，提供了多个输入选项：

随机生成输入按钮 (random\_button)。

随机数种子输入框 (random\_seed\_input)。

模型转换优化选项下拉菜单 (optimize\_input)。

模型信息文本框 (model\_info\_output)。

上传模型文件 (model\_input)。

转换后的模型文件 (converted\_model\_output)。

上传 MSlite 输入数据文件 (data\_input)。

MSlite 输入形状文本框 (ms\_shape\_input)。

推理和对比：提供了多个按钮和输出框：

运行推理按钮 (run\_infer\_button) 和推理结果文本框 (infer\_output)。

生成映射按钮 (run\_map\_button) 和映射概览文本框 (mapping\_outline)。

节点映射数据表 (model\_mapping)。

运行对比按钮 (run\_compare\_button) 和对比分析指标复选框组 (analysis\_tool)。

对比结果数据表 (compare\_output)。

绘制统计图和下载结果：

提供绘制统计图按钮 (run\_draw\_button) 和下载对比结果按钮 (dowload\_compare\_button)。

使用 @gr.render 装饰器定义了一个函数 create\_plot，用于根据对比结果绘制柱形图。

事件绑定：

绑定了多个事件处理函数：

上传模型文件时调用 process\_model 函数。

上传输入数据文件时调用 process\_input 函数。

点击随机生成输入按钮时调用 random\_input 函数。

点击运行推理按钮时调用 run\_infer 函数。

点击生成映射按钮时调用 run\_map 函数。

点击运行对比按钮时调用 run\_compare 函数。

启动界面：

最后调用 mindacc.launch() 启动 Gradio 界面。

该界面通过直观的布局和交互式组件，方便用户上传模型和输入数据，运行推理和对比分析，并查看和下载结果。

## 业务/实现流程说明

### 推理结果对比处理流程

推理结果对比的主要处理流程如下。

1.用户打开 [main.ipynb](vscode-file://vscode-app/c:/Users/23629/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.esm.html" \o ") 文件，运行所有单元格。

2.用户 打开 Gradio 界面。

3.Gradio 界面初始化 MindAccModel等。

4.用户提交模型文件。

5.Gradio 界面 加载 ONNX 模型到 MindAccModel。

6.MindAccModel 使用 ONNX Runtime 运行 ONNX 模型推理。

7.MindAccModel 使用 MindSpore Lite 运行 MindSpore Lite 模型推理。

8.MindAccModel dump 推理数据。

9.MindAccModel 将推理数据传递给 MindAccMapper 进行映射。

10.MindAccMapper 将映射后的数据提供给 MindAccAnalyzer。

11.MindAccAnalyzer 对比推理结果。

12. MindAccAnalyzer 将对比结果返回给 Gradio 界面。

13.Gradio 界面显示对比结果给用户。

图示

低可信度描述已自动生成

### 自定义映射处理流程

自定义映射处理流程大部分与2.5.1所述，当用户点击对比分析按钮之前，如果修改了UI中的映射表格，对比分析模块将自动读取修改后的映射再调用Analyzer进行分析。

## 接口描述

### 调用接口

本项目调用的主要外部接口如下所示。

**ONNX相关：**

onnx.load(self.path) - 加载 ONNX 模型。

onnx.helper.tensor\_dtype\_to\_np\_dtype(self.precision.elem\_type) - 将 ONNX 张量数据类型转换为 NumPy 数据类型。

onnxdumper.InferenceSession(self.path) - 创建推理会话。

self.onnx\_session.get\_in/outputs() - 获取 ONNX 模型的输入输出节点。

**MindSpore Lite相关：**

cmd = f"{converter} --fmk=ONNX --modelFile={self.path} --outputFile={self.path} {"--optimize=" + optimize if optimize else ""}" – 调用模型转换。

cmd=f"{benchmark.resolve()} --modelFile={str(self.path)+".ms"} --inDataFile=input/mslite\_input.bin" – 调用模型推理与数据dump。

**Gradio相关：**

调用各类组件如按钮等。

mindacc.launch() – 运行web界面并监听端口。

**Regex模块：**

filename.match(pattern) – 正则匹配提取MindSpore Lite输出。

**Logging 模块：**

使用logging进行日志分级记录。

**Numpy模块：**

加载onnx导出数据、生成随即输入、进行数据比较等。

### 提供接口

本项目提供三个类供第三方使用。

Class MindAcc Model():

load(self, model\_file: gr.File) -> None

input\_load(self, input\_data: gr.File) -> None:

run\_ms\_converter(self, optimize = "") -> Path

input\_generate(self, seed = 0) -> dict

run\_ms\_dump(self) -> Path

run\_onnx\_dump(self, dump\_path = "onnx\_dumpinrun.npz") -> dict

Class MindAccMapper:

get\_ms\_bin\_info(cls, ms\_bin\_file: Path) -> dict

read\_ms\_output(cls, ms\_output\_path: Path) -> defaultdict

simple\_map(self) -> dict

 get\_map\_result(self): -> tuple

Class MindAccAnalyzer:

analyzers={ cosine\_similarity: lambda func

relative\_euclidean\_distance: lambda func

max\_absolute\_error: lambda func

mean\_absolute\_error: lambda func

root\_mean\_square\_error: lambda func

max\_relative\_error: lambda func

mean\_relative\_error: lambda func

accumulated\_relative\_error: lambda func

standard\_deviation: lambda func

kullback\_leibler\_divergence: lambda func

}

如需扩展可以方便的调用与修改。

## UI设计

UI设计如下图所示。UI设计使用从上到下的流式布局。

标题区域：

提供简介，提供了一个链接，指向 Gitee 仓库。

模型处理模块：

左侧允许选择转换后的模型类型等。

下拉菜单中可以选择相应的量化模式或保留 none。

支持上传 ONNX 模型和 MS Lite 输入数据，或生成随机输入。

用户可以选择文件进行上传，分别上传 ONNX 模型和 MS Lite 模型输入数据。

运行推理按钮：点击后会进行模型推理。

结果展示按钮：点击后会显示推理结果的可视化。

生成对比按钮：用于生成并展示两种模型推理结果的对比分析。

结果展示区域：推理结果包括 MS Lite 输出和 ONNX 输出。

对比分析部分提供了多种误差分析选项，如 cosine\_similarity（余弦相似度）、relative\_euclidean\_distance（相对欧几里得距离）、mean\_absolute\_error（平均绝对误差）等。

对比统计与下载：

用户可以选择多种误差分析标准，并可以将对比结果下载。

图形用户界面, 应用程序, Teams

描述已自动生成

# 其他

## 成员分工

刘浩毅：担任队长，对应用进行了整体设计与代码编写，进行测试，文档撰写，发布仓库等工作。

## 困难与思考

在开发过程已经遇到并解决了许多问题，以下为我们遇到的一些困难以及产生的思考：

神经网络结构：不熟悉onnx和mindspore lite的模型结构，算子类型，需要花时间学习了解。

项目架构设计：功能接口设计需要做到高内聚低耦合，尤其是本项目作为一个工具项目，需要保证易修改，易扩展，易阅读，项目架构设计经历了好几次变化，最终差不多较为满意。

## 参考

[onnx-community (ONNX Community) (huggingface.co)](https://huggingface.co/onnx-community)

[ONNX Model Zoo](https://onnx.ai/models/)

[Netron](https://netron.app/)

[MindStudio 高精度对比随笔-云社区-华为云 (huaweicloud.com)](https://bbs.huaweicloud.com/blogs/385578?utm_source=cnblog&utm_medium=bbs-ex&utm_campaign=other&utm_content=content)

[MindSpore MindSpore Lite文档](https://www.mindspore.cn/lite/docs/zh-CN/master/index.html)

[Gradio BarPlot Docs](https://www.gradio.app/docs/gradio/barplot)

[MindSpore 精度问题初步定位指南](https://www.mindspore.cn/mindinsight/docs/zh-CN/master/accuracy_problem_preliminary_location.html)

[API Reference - ONNX 1.18.0 documentation](https://onnx.ai/onnx/api/)

[Walk through intermediate outputs - sklearn-onnx 1.18.0 documentation](https://onnx.ai/sklearn-onnx/auto_examples/plot_intermediate_outputs.html)