Introduction

Ingenuity is an optimized inference engine and benchmarking tool for TinyML models on embedded IoT devices.

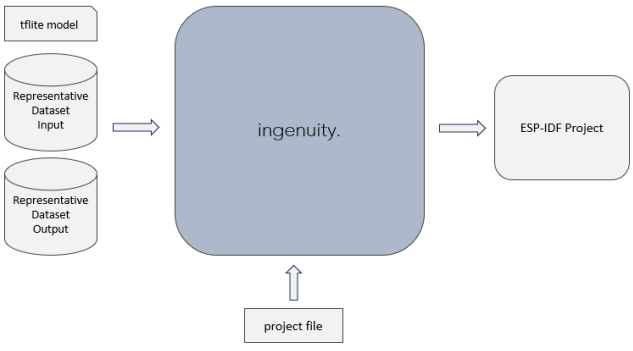
The Ingenuity **Inference Engine** is a lightweight, memory-efficient library. All buffers are pre-compiled, avoiding dynamic memory allocation, and it is optimized for high performance while maintaining a minimal, easy-to-use C API. For more information, refer to the "**Ingenuity Inference Engine**" document.

Benchmarking

Benchmarking a quantized TFLite model typically involves multiple steps, including building and deploying the model on the device, as well as designing and implementing benchmarking test suites. Ingenuity automates this entire process with a single click, seamlessly bridging the gap between model quantization and benchmarking. Through the Graphical User Interface (GUI), benchmark metrics such as inference latency, memory usage, and quantization accuracy can be easily monitored within seconds. This allows users to benchmark their models quickly and efficiently.

One-Click Process

Before execution, the project file must be properly configured with the validator's input and output representative datasets, as well as inference settings such as the inference rate and the number of inferences for benchmarking. Once configured, a single click automates the entire process—handling file generation, project building, flashing, and real-time monitoring of benchmarking results. After the benchmark is completed, the generated ESP-IDF project folder can be used to be integrated with the user's application code.



Getting-started

Prerequisites and File Preparation

Before starting the software, you need a machine learning model in **TFLite format**. The model should be **int8 quantized**, but its input and output can be either **int8 or float**. **Ingenuity** will generate an **ESP-IDF project** that includes the model, the inference engine, and a **main function** demonstrating how to use the **C API**.

If you want to **benchmark the inference engine**, you will need two additional files in **CSV format**:

- One file containing the **input values** for the model.
- Another file containing the **corresponding expected outputs**.

During benchmarking, the software will compare the model's actual output after inference with the provided expected output. This allows it to calculate the **difference** and assess the **accuracy** of the inference engine.

Graphical User Interface (GUI) Guide

Step 1: Create a new project

The **"Home"** button allows you to create a new project or load an existing one. To create a new project, follow these steps:

1. Click the **"Home"** button and select **"New Project..."**
2. In the **"New Project"** window, choose the folder where the project will be created.
3. Enter a name for the new project and click **OK**—this will open a file explorer window.
4. Locate and open the **.yaml** configuration file, then update it with the appropriate parameters. **Refer to the comments next to each value for detailed guidance.**
5. After modifying the .yaml configuration file, reload the project by clicking **"Open Project…"** from the **"Home"** button and selecting the configuration file.

```yaml
# Configuration file for the Ingenuity software.
# Note: If relative paths are used, they will be considered relative to this YAML file.
#       The 'toolchain_path' must be an absolute path.
#       The 'settings' section in this file can be modified later from the GUI.

main:
  output_directory: "esp32s3_output"        # Path where the IDF project files will be stored
  model_file: "model.tflite"                # Path to the TensorFlow Lite model file

device:
  manufacturer: "Espressif"                 # Name of the device manufacturer
  dev_model: "ESP32-S3"                     # The device model being used
  toolchain_path: "C:/Espressif_5.3.1"      # Absolute path to the ESP-IDF toolchain

validator:                                  # The csv files needed for the benchmark
  input_dataset: "rep_dataset_input.csv"    # CSV file containing the representative dataset
  output_dataset: "rep_dataset_output.csv"  # CSV file containing the expected output of the representative dataset

settings:
  generate_inference_engine: true           # Whether to generate the inference engine with the model (true or false)
  enable_benchmark: true                    # Whether to perform benchmark (true or false)
  inference_rate: 50                        # Inference interval in milliseconds (valid range: 0 to 1000 ms)
  inferences_n: 100                         # Total number of inferences to run (valid range: 1 to 10^9)
  show_graphs: true                         # Whether to display performance graphs (true or false)
```

Step 2: Update Settings (Optional)

 The **Settings** button opens the settings window, allowing you to configure benchmark parameters. **Note:** These parameters can also be modified directly in the project file before loading the project.

Step 3: Execution

 The **Execution** button starts the one-click benchmarking process. It becomes enabled after a project is loaded and consists of the following steps:

1. Generates the **ESP-IDF project**, including the **Ingenuity inference engine** library.
2. Generates the **validator files** required for benchmarking based on the input and output datasets
3. Builds the project and flashes it to the device.
4. Monitors the device output and displays the benchmark results.

After execution, the ESP-IDF project is generated, and the application code can be integrated into the project.

Benchmarking Results

The **main panel** displays benchmark results for the following metrics:

1. **Latency** – Inference latency measured in MCU cycles.
2. **Accuracy** – The accuracy of the inference engine, calculated by comparing the output with the expected output obtained from the representative dataset.
3. **Memory** –
    - The first table shows the device's overall memory usage.
    - The second table shows the inference engine’s memory usage as a separate component.
4. **Energy** – This feature is currently under development.

Supported hardware & ML models

Ingenuity supports quantized **TensorFlow Lite (TFLite) models** based on **fully connected feed-forward neural networks**. The inference engine is optimized to utilize the AI hardware accelerators and **internal** memory of the **ESP32-S3** microcontroller from **Espressif**.