Ingenuity Inference Engine

Introduction

The **Ingenuity Inference Engine** is a lightweight, memory-efficient, and high-performance library designed for running machine learning models on embedded devices.

A **TensorFlow Lite (TFLite) model** is first parsed using the tflite Python package and then converted into C buffer arrays. All buffers are **pre-compiled**, eliminating the need for dynamic memory allocation, and are optimized for high performance while maintaining a minimal, easy-to-use **C API** with no external dependencies.

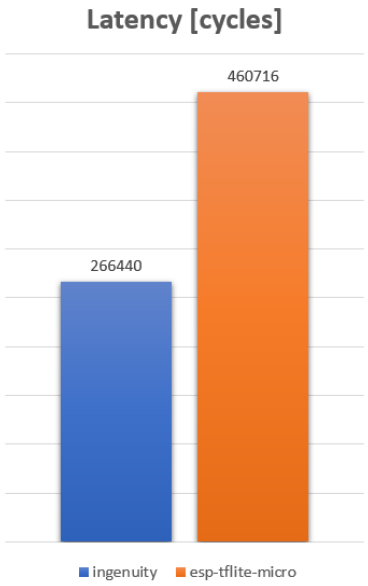
The Ingenuity Inference Engine, along with the converted model, is implemented as an **ESP-IDF component**. It is optimized for performance and leverages the AI hardware accelerators of the **ESP32-S3** microcontroller from **Espressif**. The entire component is stored in the **internal memory** of the microcontroller, ensuring low-latency execution and efficient resource utilization.

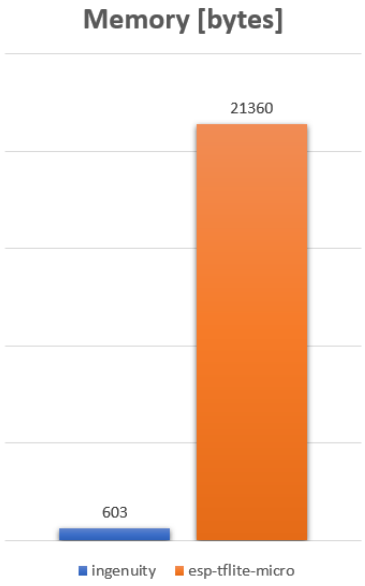
The Ingenuity Inference Engine supports quantized TensorFlow Lite models based on **fully connected feed-forward neural networks**.

Benchmark

The following benchmark serves as a demonstration and utilizes a pre-trained model from MLPerf Tiny, specifically the **Deep Autoencoder** for Anomaly Detection in machine operating sounds. The model is a quantized INT8 version with integer input and output. The comparison is conducted under identical conditions—using the same setup, hardware, input, and expected output (obtained from the TensorFlow Lite Python interpreter)—to evaluate the performance of the **Ingenuity** Inference Engine against Espressif's **ESP-TFLite-Micro**.

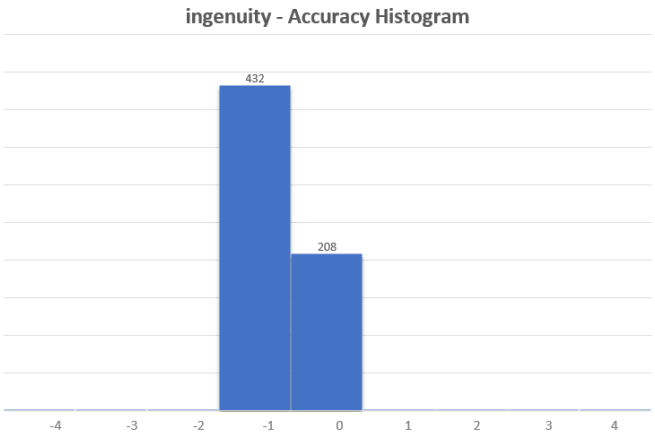
The latency for a single inference using Ingenuity is 266,440 cycles (1.665 ms), whereas with ESP-TFLite-Micro, it is 460,716 cycles (2.879 ms), both running at 160 MHz. This demonstrates a performance improvement of approximately **1.73x**.

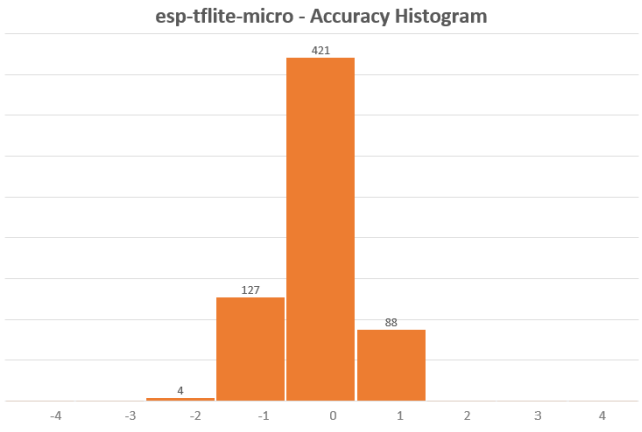




The total memory footprint is 603 bytes for Ingenuity and 21,360 bytes for ESP-TFLite-Micro, excluding the memory allocated for the model. This represents a **35.4x** reduction in memory usage.

The following histograms compare the accuracy of the two inference engines with the TFLite Python Interpreter, using an indicative input with random values. The x-axis represents the difference between the output and the expected output.





C API Usage Example

To use the C API, include the header file in your project:

```
#include "NN_lite_API.h"
```

Obtain a **pointer** to the input buffer and populate it with your data.

```
In_out_t *input_buffer = NN_lite_get_p_input();

for (int i = 0; i < NN_LITE_INPUT_LENGTH; i++)
{
    input_buffer[i] = quantized_value[i];
}
```

The model expects **int8 quantized input**, so if your data is in floating point, you must quantize it.

```
for (int i = 0; i < NN_LITE_INPUT_LENGTH; i++)
{
    input_buffer[i] = NN_lite_quantize_FloatToInt(float_value[i]);
}
```

Once the input buffer is populated, invoke the inference process.

```
NN_lite_res_t result = NN_lite_inference();

if (result != NN_LITE_SUCCESS)
{
     printf("Inference failed.\n");
}
```

After the inference completes successfully, retrieve the output buffer **pointer** and process the results.

```
In_out_t *output_buffer = NN_lite_get_p_output();

for (int i = 0; i < NN_LITE_OUTPUT_LENGTH; i++)
{
      float dequantized_value = NN_lite_dequantize_IntToFloat(output_buffer[i]);

      printf("Dequantized Output: %f\n", dequantized_value);
}
```