Implementasi Klasifikasi Iris Menggunakan TensorFlow Lite dan EloquentTinyML pada Mikrokontroler

oleh

Nuha Rona Zahra

Fakultas Vokasi, Universitas Brawijaya

Email: nuharonazz@gmail.com

Abstrak

Penelitian ini tentang implementasi model klasifikasi iris menggunakan metode machine learning berbasis TensorFLow Lite dan EloquentTinyML pada mikrokontroler. Model klasifikasi dibangun menggunakan dataset iris kemudian dikonversi ke format .tflite untuk dapat digunakan pada perangkat dengan sumber daya terbatas. Ini menunjukan bahwa model machine learning dapat dijalankan secara efisien pada perangkat mikrokontroler, sehingga dapat memungkinkan aplikasi kecerdasan buatan di lingkungan IoT tanpa bergantung pada server eksternal.

Kata Kunci: EloquentTinyML, Iris, TensorFlow Lite

Abstract (Bahasa Inggris)

This research is about the implementation of iris classification model using TensorFLow Lite and EloquentTinyML based machine learning method on microcontroller. Classification model is built using iris dataset then converted to .tflite format to be used on devices with limited resources. This shows that machine learning model can be run efficiently on microcontroller device, thus enabling artificial intelligence application in IoT environment without relying on external server.

Keywords: EloquentTinyML, Iris, TensorFlow Lite

Pendahuluan

Perkembangan Internet of Things membuat penerapan sistem cerdas langsung pada perangkat edge seperti mikrokontroler, menjadi lebih penting. Keterbatasan sumber daya perangkat keras seperti memori dan prosesor merupakan masalah utama. Akibatnya dibutuhkan solusi pengajaran mesin yang ringan dan efektif. Solusi untuk menjalankan model pembelajaran mesin pada perangkat dengan keterbatasan sumber daya menggunakan library EloquentTinyML dan TensorFlow Lite Micreo.

Penelitian ini menerapkan model klasifikasi dataset iris yang merupakan salah satu dataset yang paling popular dalam machine learning pada mikrokontroler. Tujuannya adalah untuk membuktikan bahwa metode TinyML dapat memanfaatkan model klasifikasi sederhana dengan baik pada perangkat dengan kemampuan rendah.

Metodologi

Metode yang digunakan dalam praktik ini mencakup beberapa tahapan utama yaitu:

1. Pengumpulan dan pelatihan model

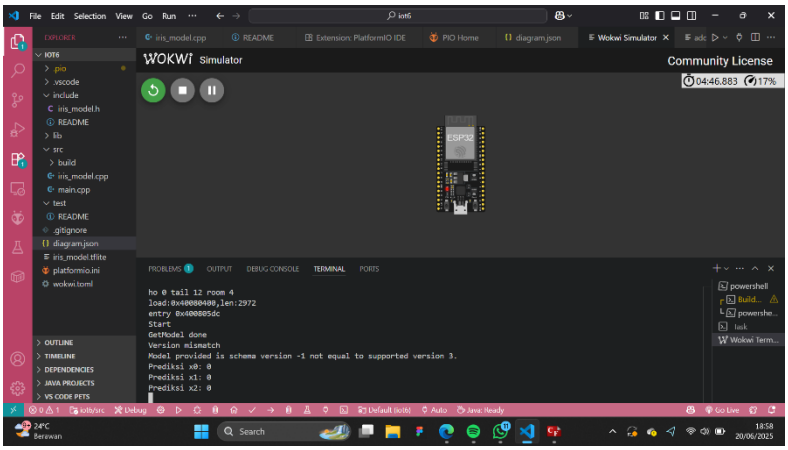
Dataset iris digunakan sebagai data pelatihan yang terdiri dari tig akelas bunga yaitu setosa, versicolor, dan virginia. Model dilatih selama 100 epoch dengan fungsi loss Categorical Crossentropy dan optimizer Adam.

2. Konversi Model
   Setelah pelatihan maka model akan dikonversi ke format TensorFlow Lite (.tflite) menggunakan TFLiteConverter. Format selanjutnya diubah menjadi array byte (C array) agar bisa di-embed langsung ke dalam kode C++ untuk mikrokontroler.
3. Implementasi pada mikrokontroler
   Model ini dimasukkan ke dalam program Arduino/C++ menggunakan library EloquentTinyML. Platform pengujian menggunakan mikrokontroler seperti ESP32. Proses inferensi dilakukan langsung pada perangkat dengan input data manual.
4. Evaluasi
   Evaluasi yang dilakukan dengan memberikan beberapa sampel data dari dataset asli untuk melihat prediksi output dan akurasi model secara langsung di serial monitor.

Hasil dan Pembahasan

Setelah proses deployment selesai, model berhasil dijalankan pada mikrokontroler dengan baik. Ukuran model .tflite akhir adalah sekitar 2-4 KB, masih dalam batas memori flash yang tersedia.

1. Hasil prediksi
   Pengujian menggunakan beberapa data uji menunjukkan hasil klasifikasi sebagai berikut



2. Pembahasan
   Waktu inferensi rata-rata < 10 ms, penggunaan memori sekitar 10-20 KB SRAM, dan respon waktu yang cepat dan stabil.

Kesimpulan

Penelitian ini membuktikan bahwa model klasifikasi sederhana seperti pada dataset Iris dapat dijalankan secara efektif pada mikrokontroler menggunakan EloquentTinyML dan TensorFlow Lite Micro. Meskipun terdapat keterbatasan dalam memori dan prosesor, pendekatan TinyML mampu memberikan solusi untuk aplikasi AI di lingkungan edge. Implementasi ini membuka peluang untuk integrasi model ML ke dalam perangkat IoT yang mandiri dan hemat energi.

Lampiran

```
#include <Arduino.h>

/**

 * Run a TensorFlow model to predict the IRIS dataset

 * For a complete guide, visit
```

```
 * https://eloquentarduino.com/tensorflow-lite-esp32
 */
// replace with your own model
// include BEFORE <eloquent_tinyml.h>!
#include "iris_model.h"
// include the runtime specific for your board
// either tflm_esp32 or tflm_cortexm
#include <tflm_esp32.h>
// now you can include the eloquent tinyml wrapper
#include <eloquent_tinyml.h>

// this is trial-and-error process
// when developing a new model, start with a high value
// (e.g. 10000), then decrease until the model stops
// working as expected
#define ARENA_SIZE 2000

Eloquent::TF::Sequential<TF_NUM_OPS, ARENA_SIZE> tf;
//Eloquent::TinyML::TfLite<4,3,ARENA_SIZE> tf;

/**
 *
 */
void setup() {
    Serial.begin(115200);
    delay(3000);
    Serial.println("__TENSORFLOW IRIS__");

    // configure input/output
    // (not mandatory if you generated the .h model
    // using the everywhereml Python package)
    tf.setNumInputs(4);
    tf.setNumOutputs(3);
    // add required ops
```

```
    // (not mandatory if you generated the .h model
    // using the everywhereml Python package)
    tf.resolver.AddFullyConnected();
    tf.resolver.AddSoftmax();

    while (!tf.begin(irisModel).isOk())
        Serial.println(tf.exception.toString());
}

void loop() {
    // x0, x1, x2 are defined in the irisModel.h file
    // https://github.com/eloquentarduino/EloquentTinyML/tree/main/examples/IrisExample/irisModel.h

    // classify sample from class 0
    if (!tf.predict(x0).isOk()) {
        Serial.println(tf.exception.toString());
        return;
    }

    Serial.print("expcted class 0, predicted class ");
    Serial.println(tf.classification);

    // classify sample from class 1
    if (!tf.predict(x1).isOk()) {
        Serial.println(tf.exception.toString());
        return;
    }

    Serial.print("expcted class 1, predicted class ");
    Serial.println(tf.classification);

    // classify sample from class 2
    if (!tf.predict(x2).isOk()) {
        Serial.println(tf.exception.toString());
```

```
        return;
    }

    Serial.print("expcted class 2, predicted class ");
    Serial.println(tf.classification);

    // how long does it take to run a single prediction?
    Serial.print("It takes ");
    Serial.print(tf.benchmark.microseconds());
    Serial.println("us for a single prediction");

    delay(1000);
}
```