Minor in AI

TinyML

Deployment of Quantized TFLite Model on ESP32

June 10, 2025

1 Introduction

Tiny Machine Learning (TinyML) represents a paradigm shift in the deployment of machine learning models, enabling inference on low-power, resource-constrained devices such as microcontrollers. One of the key enablers of this technology is TensorFlow Lite, a lightweight version of TensorFlow designed for edge devices.

This document outlines the complete process of developing, quantizing, and deploying a TensorFlow Lite (TFLite) model trained on the Iris dataset to an ESP32 microcontroller. The workflow includes dataset preprocessing, neural network construction, model quantization, conversion to TFLite, and embedded deployment using the Arduino IDE. Special attention is given to evaluating inference performance and model footprint to assess the feasibility of real-time, on-device learning.

2 Building and Training the Neural Network

The session began with importing necessary packages such as `numpy`, `tensorflow`, `pandas`, and `sklearn`. The classic Iris dataset was used as a base, featuring four input features—sepal length, sepal width, petal length, and petal width—corresponding to three flower species.

The dataset was split into training (80%) and test (20%) sets using `train_test_split`. A simple feedforward neural network was created using TensorFlow's Sequential API:

- **Input layer:** 4 neurons
- **Hidden layer:** 10 neurons with ReLU activation
- **Output layer:** Softmax activation for 3-class prediction

The model was compiled with the Adam optimizer, sparse categorical cross-entropy loss, and accuracy as the metric. Training for 15 epochs yielded up to 100% training accuracy, demonstrating the power of even a small neural network on clean datasets.

3 Model Quantization and Conversion

To prepare for deployment, the trained Keras model was converted to the TensorFlow Lite (TFLite) format using the TFLiteConverter API. Quantization from float32 to float16 was applied to reduce the model size and improve performance. The result was a compact **2KB** TFLite file, highly suitable for embedded systems.

4 Testing the TFLite Model in Colab

Using the TFLite interpreter in Google Colab, a sample input was passed to the quantized model for inference. The input/output tensor details were printed, and the prediction probabilities across the three classes were analyzed. Inference time was measured in microseconds, highlighting the model's efficiency.

Example Output

Predicted probabilities: [0.02, 0.95, 0.03]
Predicted class: 1
Inference time: 152 μs

5 Preparing for Hardware Deployment

To deploy on ESP32, the quantized TFLite model was transformed into a C-style header file using Python code. The file consisted of an `unsigned char` array representing model bytes in hexadecimal.

```
const unsigned char model[] = {
  0x20, 0x00, 0x00, 0x00, ...
};
```

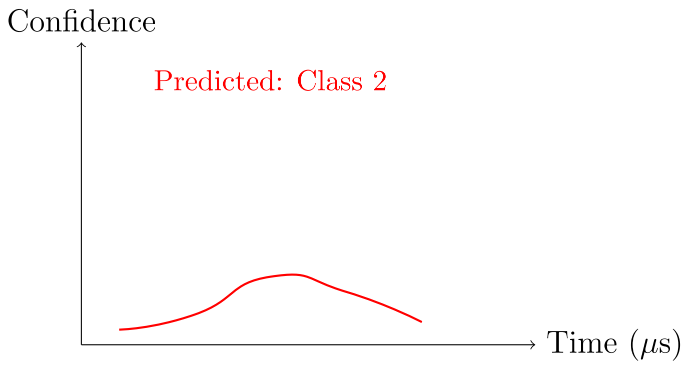
This header was included in an Arduino sketch, ready for deployment.

6 Arduino IDE Setup and ESP32 Programming

The Arduino IDE was used for loading the sketch to the ESP32 board. Necessary libraries, including TensorFlow Lite for Arduino, were installed. Key steps included:

- Initializing the interpreter with a 2KB tensor arena
- Feeding hard-coded input features
- Running inference and printing output probabilities

Hardware Inference Visualization



7 Deploying and Running the Model on ESP32

After selecting the appropriate board (e.g., ESP32 Dev Module) and COM port, the sketch was uploaded. On successful flashing, the serial monitor displayed class probabilities and inference time, e.g., `140`μ`s`. This confirmed that even small devices like ESP32 are capable of running ML models in real time.

Special care was taken for hardware debugging, especially for Windows users who needed to press the boot button during flashing. The instructor emphasized that hardware deployments are sensitive to many real-world factors such as loose cables, power supply issues, and inconsistent timing.

Key Takeaways

1. A small neural network trained on the Iris dataset can be effectively quantized and deployed.
2. TensorFlow Lite enables model conversion to highly efficient formats suitable for edge devices.
3. Quantized models are tiny in size (2KB) and achieve microsecond-level inference.
4. Arduino IDE, along with ESP32, provides a beginner-friendly platform for deploying TinyML models.
5. Real-world deployment involves translating model logic into embedded C/C++, configuring memory manually, and verifying performance using serial monitoring.
6. Hands-on practice, including converting models and writing C header files, is key to mastering TinyML deployment.