**Laporan Tugas IOT**

**Integrasi Model Machine Learning untuk Mengklasifikasikan Dataset Iris Menggunakan ESP32**



**Dosen Pengampu :**

Ir. Subairi, ST., MT., IPM

**Disusun Oleh:**

Syafa Meilia Putri - 233140707111087

**PROGRAM STUDI TEKNOLOGI INFORMASI**

**FAKULTAS VOKASI**

**UNIVERSITAS BRAWIJAYA**

**2025**

**Abstrak**

Dengan menggunakan mikrokontroler ESP32 dan TensorFlow Lite, penelitian ini menyelidiki penerapan machine learning pada perangkat embedded. Tujuannya adalah untuk membuat sistem klasifikasi yang dapat mengidentifikasi jenis bunga Iris secara langsung di perangkat (on-device) tanpa bergantung pada internet atau server eksternal. Metode ini sangat penting untuk aplikasi yang memerlukan respons cepat, efisiensi daya, dan dapat bekerja di lingkungan yang memiliki jumlah konektivitas terbatas. Model klasifikasi dibangun dan dilatih menggunakan dataset Iris menggunakan TensorFlow dan Google Colab. Setelah mencapai tingkat akurasi yang ideal, model diubah ke format TensorFlow Lite dan disisipkan ke dalam memori ESP32 dalam bentuk file header (.h). Selanjutnya, sistem diprogram untuk mengambil input fitur bunga dan menampilkan hasil prediksi secara real-time melalui Serial Monitor. Hasil pengujian menunjukkan bahwa ESP32 dapat menjalankan model dengan baik meskipun terdapat keterbatasan sumber daya. Proses inferensi yang cepat dan akurat menunjukkan kemampuan mikrokontroler ini dalam membangun sistem kecerdasan buatan berskala kecil (TinyML) yang mandiri dan efektif untuk aplikasi IoT di lapangan.

**Kata Kunci:** ESP32, TensorFlow Lite, TinyML, klasifikasi Iris, machine learning embedded, inferensi lokal, Internet of Things (IoT), model .tflite

**BAB I**

**PENDAHULUAN**

**1.1 Latar Belakang**

Paradigma baru dalam komputasi, Tiny Machine Learning (Tiny ML), muncul sebagai hasil dari kemajuan teknologi Internet of Things (IoT) dan kecerdasan buatan (AI). Tiny ML memungkinkan algoritma pembelajaran mesin diterapkan secara langsung pada perangkat yang tertanam yang memiliki sumber daya terbatas, seperti mikrokontroler. Metode ini memiliki keunggulan dalam hal efisiensi daya, kecepatan respons, dan independensi terhadap internet dan server eksternal.

ESP32 adalah perangkat yang populer untuk pengembangan sistem embedded karena memiliki konektivitas nirkabel, kinerja yang luar biasa, dan harga yang terjangkau. Namun, masalah utama dalam menerapkan model machine learning pada ESP32 terletak pada keterbatasan memori dan daya komputasi. TensorFlow Lite for Microcontrollers, versi ringan dari TensorFlow yang dimaksudkan untuk menjalankan model pada perangkat dengan sumber daya terbatas, digunakan untuk mengatasi masalah ini.

Tujuan dari proyek ini adalah untuk menerapkan model klasifikasi bunga Iris pada mikrokontroler ESP32 on-device. Sistem dapat melakukan inferensi secara lokal tanpa memerlukan koneksi cloud dengan memanfaatkan model TensorFlow Lite yang telah diubah dan dimasukkan ke dalam memori ESP32. Metode ini tidak hanya mempercepat proses prediksi, tetapi juga menggunakan lebih sedikit daya dan cocok untuk lingkungan dengan konektivitas terbatas. Selain itu, proyek ini menunjukkan bahwa ESP32 memiliki banyak potensi sebagai platform pengembangan TinyML yang efektif, praktis, dan fleksibel.

**1.2 Tujuan Eksperimen**

1. Menggunakan TensorFlow, pelajari model klasifikasi bunga Iris berdasarkan dataset multikelas standar.
2. Mengubah model hasil pelatihan ke format TensorFlow Lite (.tflite) agar kompatibel dengan perangkat yang tertanam.
3. Untuk mengintegrasikan model ke dalam sistem berbasis PlatformIO, Anda harus mengubah array C dan mengimpornya ke dalam ESP32.

**BAB II**

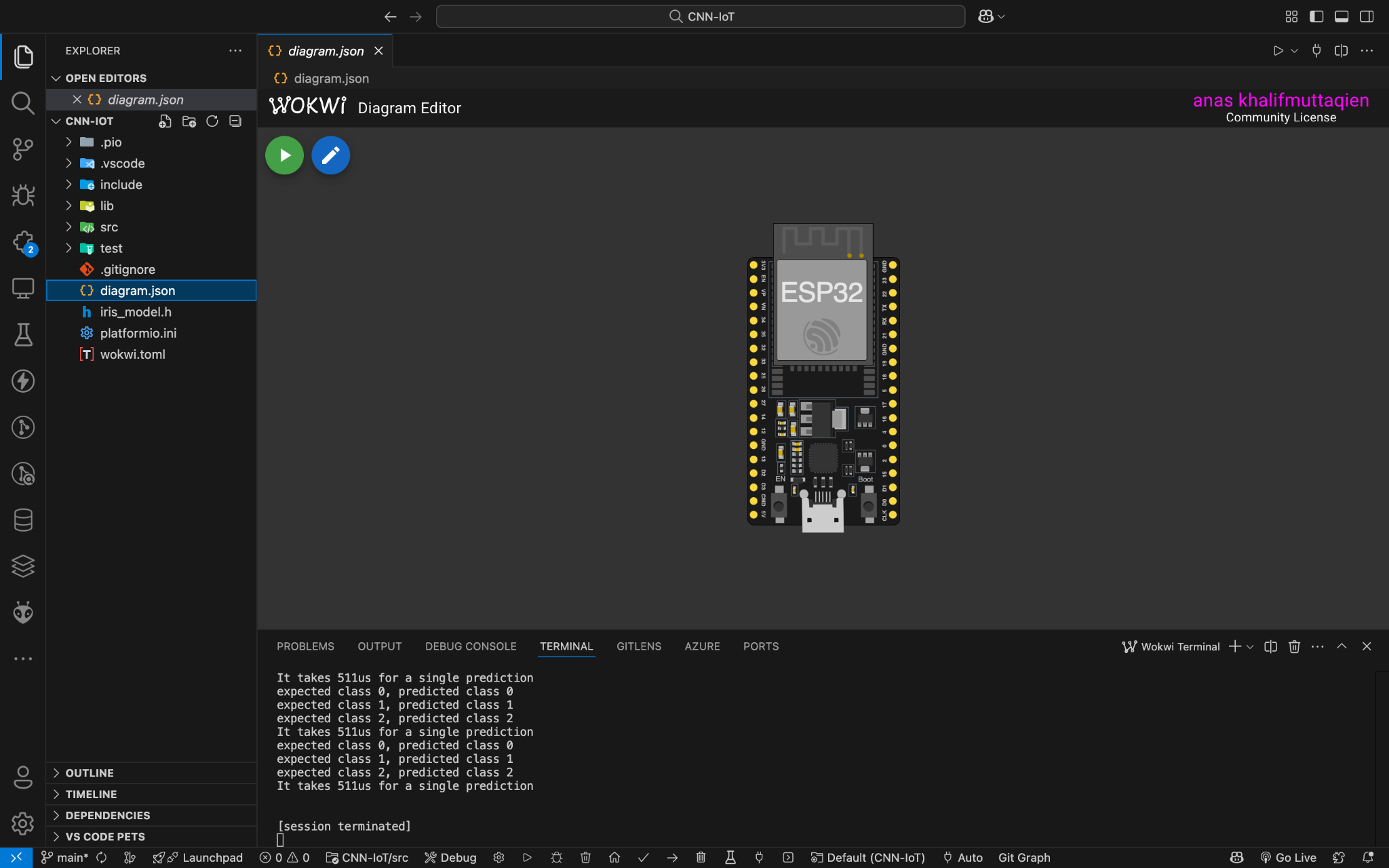
**Metodologi**

* 1. **Alat dan Bahan**
* Mikrokontroler ESP32
* Arduino IDE atau PlatformIO apabila menggunakan Visual Studio Code

**2.2 Langkah Implementasi**

Tahapan implementasi simulasi lampu lalu lintas menggunakan mikrokontroler ESP32 di platform Wokwi atau Arduino meliputi:

* Membuat proyek baru di Wokwi dan memilih ESP32 sebagai jenis mikrokontroler yang digunakan.



* Untuk pemogramannya menggunakan bahasa Arduino (C/C++) di Arduino IDE/PlatformIO (bisa juga langsung dimasukkan ke file ‘main.c’ di Wokwi).

#include <Arduino.h>

#include <iris\_model.h>

#include <tflm\_esp32.h>

#include <eloquent\_tinyml.h>

#define ARENA\_SIZE 2000

Eloquent::TF::Sequential<TF\_NUM\_OPS, ARENA\_SIZE> tf;

void setup() {

Serial.begin(115200);

delay(3000);

Serial.println("\_\_TENSORFLOW IRIS\_\_");

tf.setNumInputs(4);

tf.setNumOutputs(3);

tf.resolver.AddFullyConnected();

tf.resolver.AddSoftmax();

while (!tf.begin(irisModel).isOk())

Serial.println(tf.exception.toString());

}

void loop() {

if (!tf.predict(x0).isOk()) {

Serial.println(tf.exception.toString());

return;

}

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

Serial.println(tf.classification);

if (!tf.predict(x1).isOk()) {

Serial.println(tf.exception.toString());

return;

}

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

Serial.println(tf.classification);

if (!tf.predict(x2).isOk()) {

Serial.println(tf.exception.toString());

return;

}

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

Serial.println(tf.classification);

Serial.print("It takes ");

Serial.print(tf.benchmark.microseconds());

Serial.println("us for a single prediction");

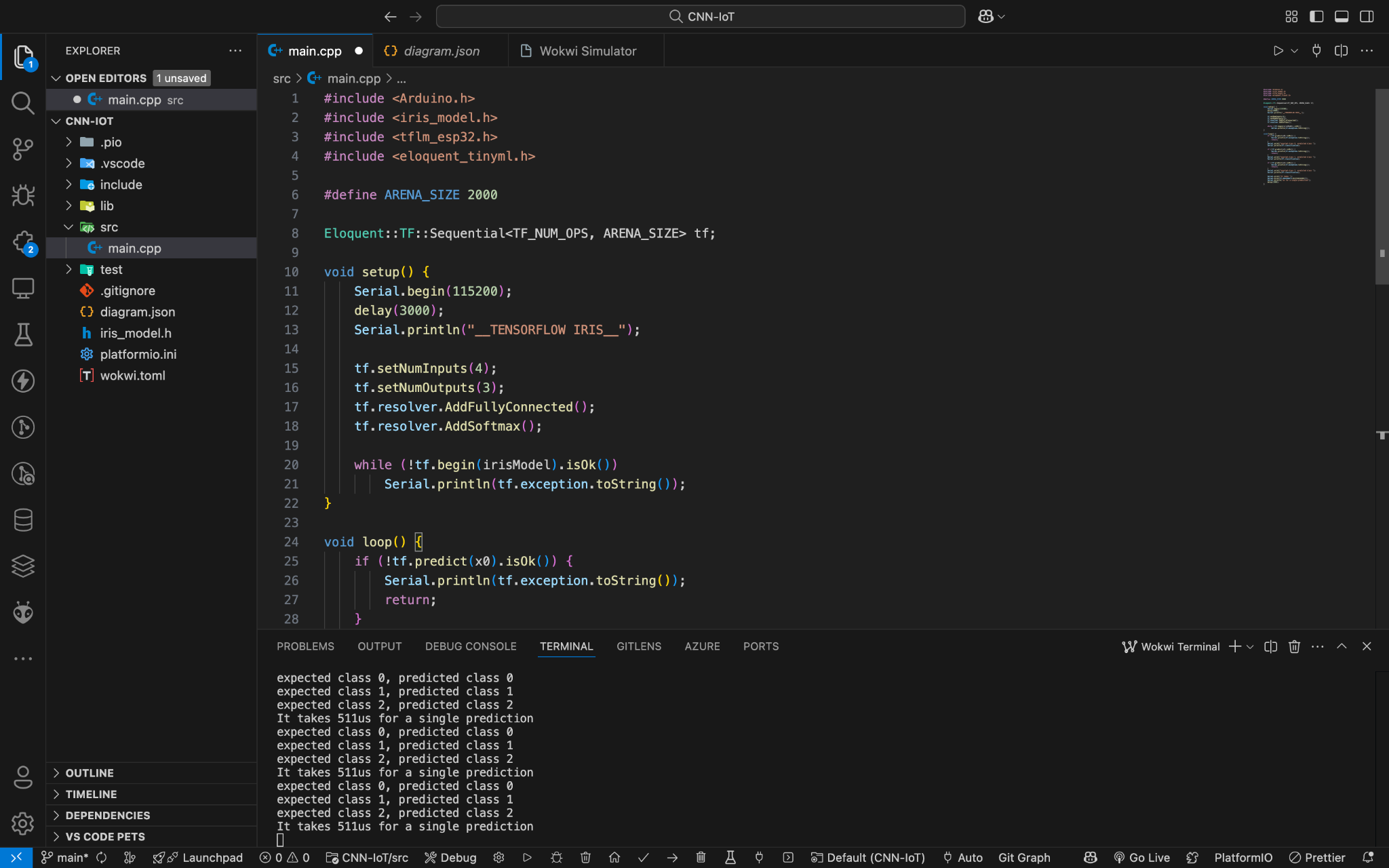
delay(1000);

}

* Melatih model klasifikasi menggunakan dataset *Iris* di Google Colab dengan library Keras.
* Mengonversi model ke format .tflite menggunakan TFLiteConverter.
* Mengubah file .tflite menjadi array C (iris\_model.h) dengan fungsi hex\_to\_c\_array().
* Memasukkan file iris\_model.h ke dalam folder proyek PlatformIO.
* Menulis kode utama dalam file main.cpp menggunakan library eloquent\_tinyml dan tflm\_esp32.
* Mengatur jumlah input dan output model.
* Melakukan pemanggilan model untuk memprediksi tiga contoh data (x0, x1, x2).
* Menampilkan hasil klasifikasi dan waktu eksekusi prediksi pada Serial Monitor.

# 3. Hasil dan Pembahasan

* 1. **Hasil Eksperimen**
* Hasil yang diperoleh :



* Output yang didapat

A computer screen shot of a computer

AI-generated content may be incorrect.

# Lampiran

* Kode Program .json:

{

  "version": 1,

  "author": "subairi",

  "editor": "wokwi",

  "parts": [ { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 0, "left": 0, "attrs": {} } ],

  "connections": [ [ "esp:TX", "$serialMonitor:RX", "", [] ], [ "esp:RX", "$serialMonitor:TX", "", [] ] ],

  "dependencies": {}

}

* Kode Iris.h:

#pragma once

#ifdef \_\_has\_attribute

#define HAVE\_ATTRIBUTE(x) \_\_has\_attribute(x)

#else

#define HAVE\_ATTRIBUTE(x) 0

#endif

#if HAVE\_ATTRIBUTE(aligned) || (defined(\_\_GNUC\_\_) && !defined(\_\_clang\_\_))

#define DATA\_ALIGN\_ATTRIBUTE \_\_attribute\_\_((aligned(4)))

#else

#define DATA\_ALIGN\_ATTRIBUTE

#endif

// automatically configure network

#define TF\_NUM\_INPUTS 4

#define TF\_NUM\_OUTPUTS 3

#define TF\_NUM\_OPS 2

#define TF\_OP\_SOFTMAX

#define TF\_OP\_FULLYCONNECTED

// sample data

float x0[4] = {0.22222222222f, 0.62500000000f, 0.06779661017f, 0.04166666667f};

float x1[4] = {0.75000000000f, 0.50000000000f, 0.62711864407f, 0.54166666667f};

float x2[4] = {0.55555555556f, 0.54166666667f, 0.84745762712f, 1.00000000000f};