CEC Continuing Education Center

DesignNews

Getting Started in TinyML with Arduino

DAY 4: Build a Classification Model: Fruit to Emoji Project

Sponsored by

DigiKey

informa markets

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.

Dr. Don Wilcher

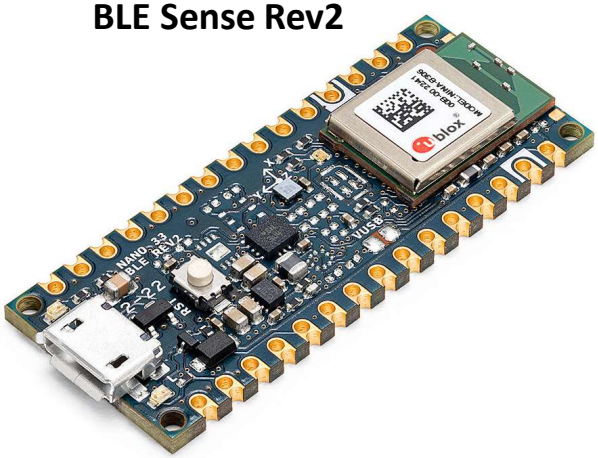
Visit ‘Lecturer Profile’ in your console for more details.

LinkedIn Page:
https://www.linkedin.com/in/dr-don-wilcher-ed-d-mseit-ee-ceta-2735151/

Patreon Page:
https://www.patreon.com/c/DrDon683

Course Kit and Materials

Arduino Nano 33 BLE Sense Rev2



Research Literature and Documentation

Tiny Machine Learning: Progress and Futures

Ji Lin Ligeng Zhu Wei-Ming Chen Wei-Chen Wang Song Han

Massachusetts Institute of Technology

https://tinyml.mit.edu

Abstract—Tiny Machine Learning (TinyML) is a new frontier of machine learning. By squeezing deep learning models into billions of IoT devices and microcontrollers (MCUs), we expand the scope of AI applications and enable ubiquitous intelligence. However, TinyML is challenging due to hardware constraints: the tiny memory resource makes it difficult to hold deep learning models designed for cloud and mobile platforms. There is also limited compiler and inference engine support for bare-metal devices. Therefore, we need to co-design the algorithm and system stack to enable TinyML. In this review, we will first discuss the definition, challenges, and applications of TinyML. We then survey the recent progress in TinyML and deep learning on MCUs. Next, we will introduce MCUNet, showing how we can achieve ImageNet-scale AI applications on IoT devices with *system-algorithm co-design*. We will further *extend* the solution from *inference to training* and introduce tiny on-device training techniques. Finally, we present future directions in this area. Today's "large" model might be tomorrow's "tiny" model. The scope of TinyML should evolve and adapt over time.

Index Terms—TinyML, Efficient Deep Learning, On-Device Training, Learning on the Edge

I. Overview of Tiny Machine Learning

Machine learning (ML) has made significant impacts on various fields, including vision, language, and audio. However, state-of-the-art models often come at the cost of high computation and memory, making them expensive to deploy. To address this, researchers have been working on efficient algorithms, systems, and hardware to reduce the cost of machine learning models in various deployment scenarios. There are two main subdomains of efficient ML: EdgeML and CloudML (Figure 1). While CloudML focuses on improving latency and throughput on cloud servers, EdgeML focuses on improving energy efficiency, latency, and privacy on edge devices. These two domains also intersect in areas such as hybrid inference [1, 2], over-the-air (OTA) updates, and federated learning between the edge and cloud [3]. In recent years, there has been significant progress in extending the scope of EdgeML to ultra-low-power devices such as IoT devices and microcontrollers, known as **TinyML**.

TinyML has several key advantages. It enables machine learning using only a few hundred kilobytes of memory which greatly reduces the cost. With billions of IoT devices producing more and more data in our daily lives, there is a growing need for low-power, always-on, on-device AI. By performing on-device inference near the sensor, TinyML enables better responsiveness and privacy while reducing the energy cost associated with wireless communication. On-device processing of data can be beneficial for applications where real-time decision-making is crucial, such as autonomous vehicles.

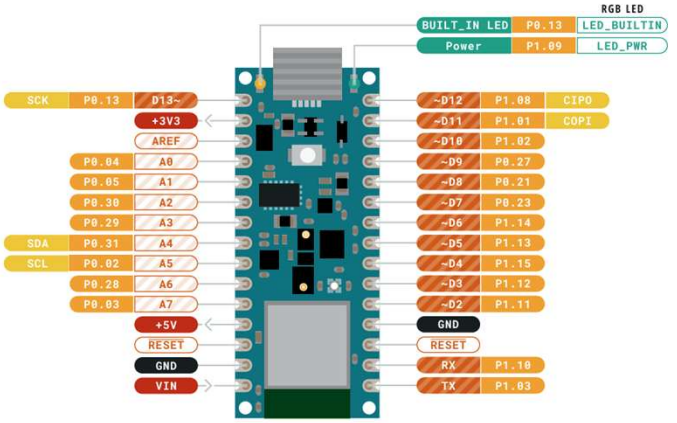
Fig. 1. Efficiency is critical for CloudML, EdgeML, and TinyML. CloudML targets high-throughput accelerators like GPUs, while EdgeML focuses on portable devices like mobile phones. TinyML further pushes the efficiency boundary, enabling powerful ML models to run on ultra-low-power devices such as microcontrollers.

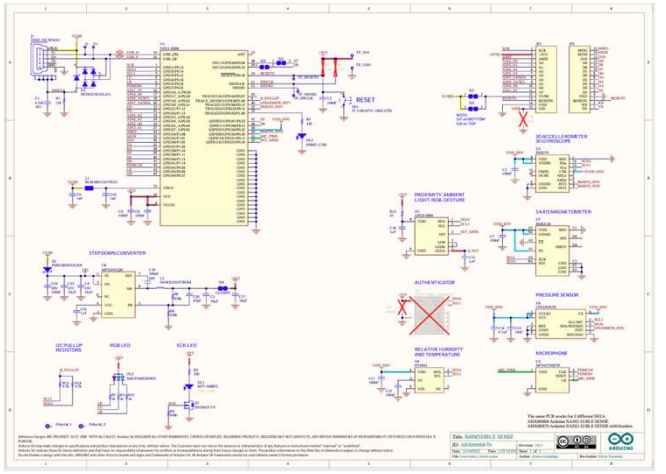
In addition to inference, we push the frontier of TinyML to enable on-device training on IoT devices. Irrevolutionizes EdgeAI through continuous and lifelong learning. Edge device can finetune the model on itself rather than transmitting data to cloud servers, which protects privacy. On-device learning has numerous benefits and a variety of applications. For example, home cameras can continuously recognize new faces, and email clients can gradually improve their prediction by updating customized language models. It also enables IoT applications that do not have a physical connection to the internet to adapt to the environment, such as precision agriculture and ocean sensing.

In this review, we will first discuss the definition and challenges of TinyML, analyzing why we can't directly scale mobile ML or cloud ML models for tinyML. Then we delve into the importance of system-algorithm co-design in TinyML. We will then survey recent literature and the progress of the field, presenting a holistic survey and comparison in Tables 1 and 10. Next, we will introduce our TinyML project, MCUNet, which combines efficient system and algorithm design to enable TinyML for both inference to training. Finally, we will discuss several emerging topics for future research directions in the field.

A. Challenges of TinyML

The success of deep learning models often comes at the cost of high computation, which is not feasible for use in TinyML applications due to the strict resource constraints of devices such as microcontrollers. Deploying and training AI models on MCU is extremely hard: No DRAM, no operating systems (OS), and strict memory constraints (SRAM is smaller than 256kB, and FLASH is read-only). The available resources on these devices are orders of magnitude smaller than those





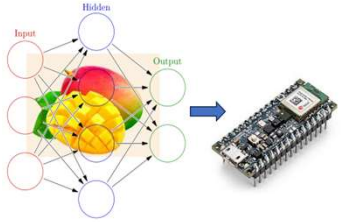
Research Perspective

"Today's large model might be tomorrow's *tiny model*."
[1] Lin et al., 2024

Agenda:

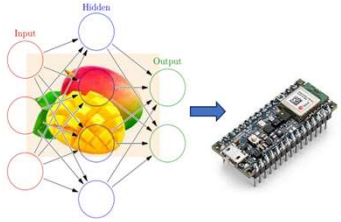
- What is a Neural Network?
- What is Keras and Pandas?
- Why the Fruit to Emoji project Is Important?
- Lab: Building a Fruit to Emoji Classifier

What is a Neural Network?



- A neural network is a computational model inspired by the way the human brain processes information.
- It is designed to recognize patterns, make decisions, and learn from data by adjusting internal parameters.

What is a Neural Network?. . .



Basic Concept

A neural network consists of layers of interconnected nodes (called *neurons*). Each neuron receives inputs, processes them, and passes an output to the next layer.

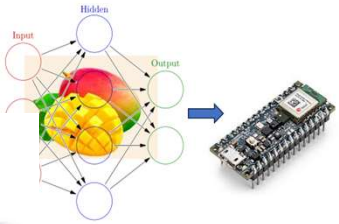
Think of it like a system that:

a) Takes in data

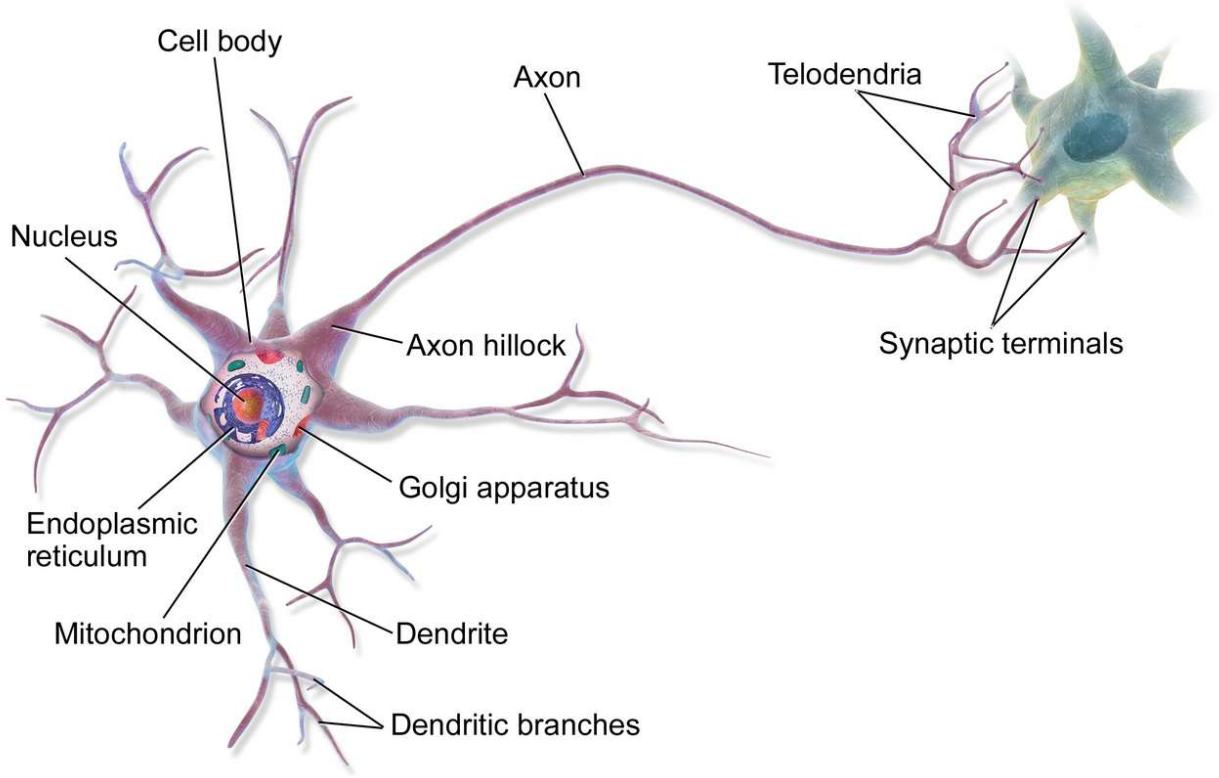
b) Applies mathematical transformations

c) Learns to improve its outputs by adjusting internal weights.

What is a Neural Network?. . .



Multipolar Neuron



BruceBlasen,Wikipedia

Question 1

What is a neural network?

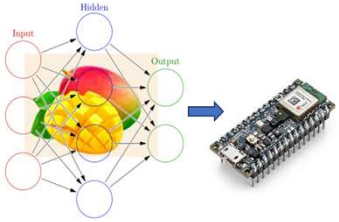
a) A computational model inspired by robotic processes.

b) A computational model inspired by the way the human brain processes noise.

c) A computational model inspired by the way the human brain processes information.

d) none of the above

What is a Neural Network?



Structure of a Neural Network

a. Input Layer

- Receives the raw data
- Example: sensor readings, pixels from an image, words in a sentence

b. Hidden Layers

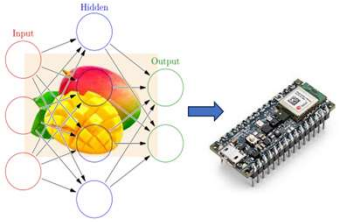
- Perform mathematical transformations
- Can be 1 layer (simple networks) or dozens (deep learning)
- Each neuron performs:

  output = activation(weighted_sum(inputs) + bias)

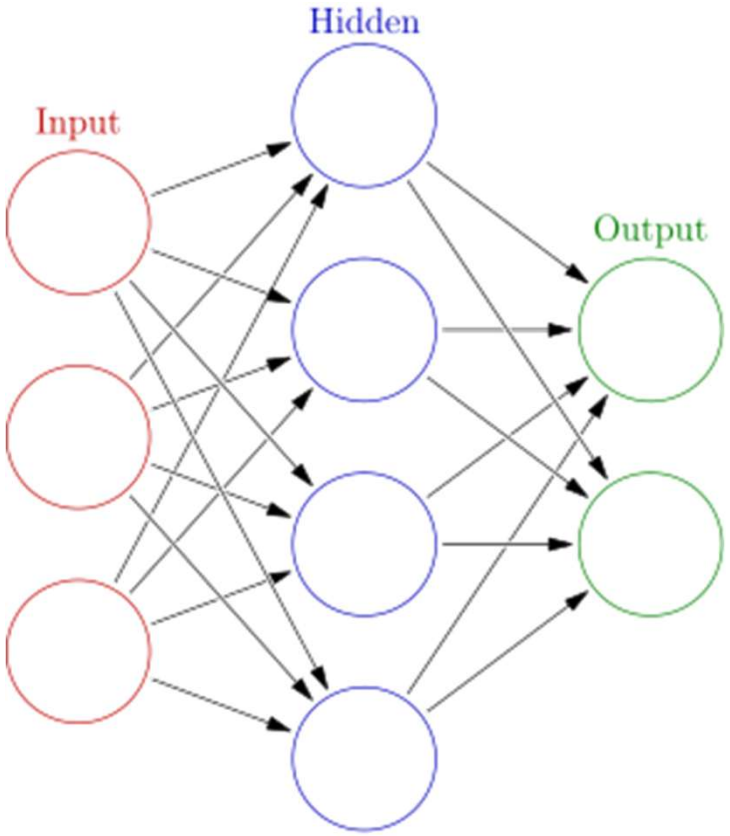
c. Output Layer

- Produces the final prediction
- Example: “apple,” “orange,” 0.87 probability of spam, a steering command for a robot.

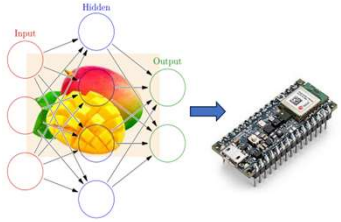
What is a Neural Network?. . .



Example Neural Network



What is a Neural Network?. . .



How a Neural Network Learns

Neural networks learn through a process called training, usually involving:

a) Forward Propagation

Data moves through the network to generate a prediction.

b) Loss Calculation

The network measures how far the prediction is from the correct answer.

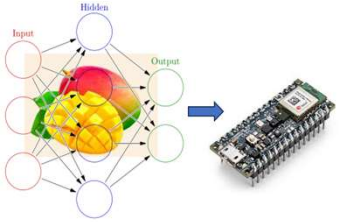
c) Backpropagation

Adjusts the network's weights to reduce future errors.

d) Optimization

Algorithms like SGD, Adam, or RMSprop improve the learning process.

What is a Neural Network?. . .



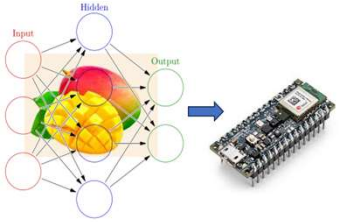
Why Neural Networks Are Powerful?

They can learn **complex, nonlinear relationships** in data that traditional programming cannot easily model.

They excel in:

a) Image classification
b) Speech recognition
c) Robotics control
d) Predictive maintenance
e) Natural language processing
f) Time-series forecasting

What is a Neural Network?. . .



Types of Neural Networks

Some common architectures:

a) Feedforward Neural Networks

Basic type, data moves in one forward direction.

b) Convolutional Neural Networks (CNNs)

Used for images and visual recognition.

c) Recurrent Neural Networks (RNNs) & LSTMs

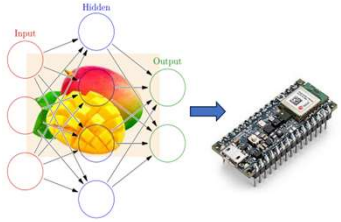
Used for sequences—speech, text, sensor signals.

Question 2

Which approach is incorrect for how a neural network learns?

a) Reverse Propagation
b) Forward Propagation
c) Back Propagation
d) Loss Calculation

What is a Neural Network?. . .



d) Transformers

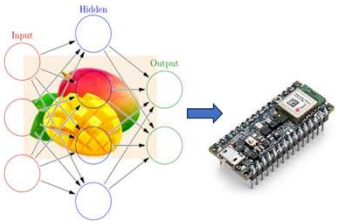
Current state-of-the-art in NLP and vision (used by ChatGPT).

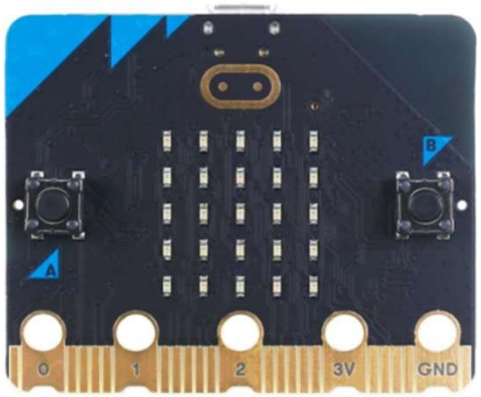
f) TinyML Models

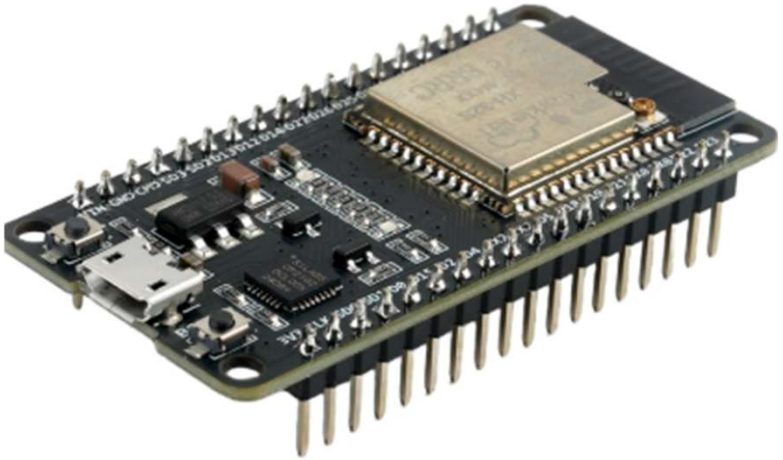
Optimized small neural networks that run on microcontrollers like:

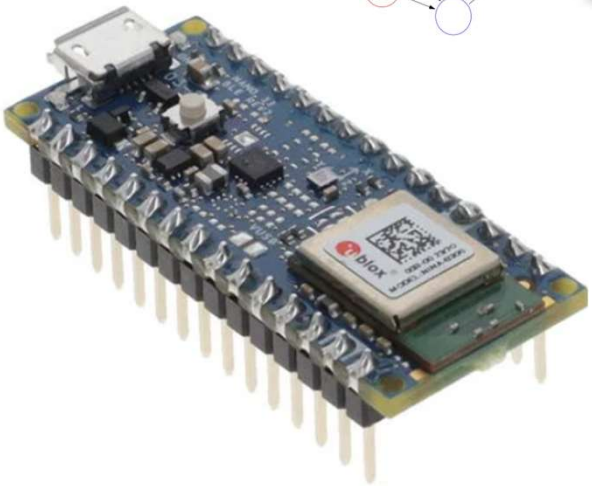
- Arduino Nano 33 BLE Sense
- ESP32
- micro:bit

What is a Neural Network?. . .

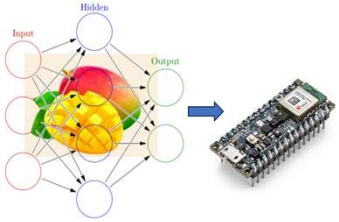








What is Keras and Pandas?. . .



What is Keras?

Keras is a high-level deep learning library that makes it easy:

a) to build, train, and deploy neural networks.

b) It runs on top of powerful machine-learning engines like TensorFlow.

Other elements related to Keras are:

a) A user-friendly neural network API

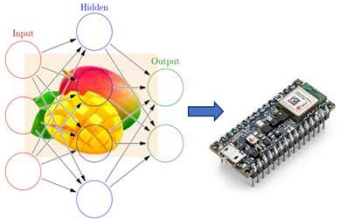
b) Written in Python

c) Designed for fast experimentation

d) Used for building deep learning models with just a few lines of code.

Note: It hides much of the complexity of TensorFlow, letting the ML developer to focus on designing and training models—not the low-level details.

What is Keras and Pandas?



What is Keras?

Models created in Keras can be exported to:

a)TensorFlow Lite

b)TensorFlow Lite Micro for deployment on microcontrollers like:

c) Arduino Nano 33 BLE Sense

d) ESP32

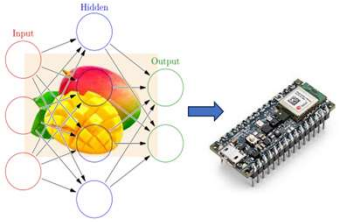
e) Raspberry Pi Pico

Question 3

What is Keras?

a) A high-level deep learning program.
b) A high-level algorithm.
c) A high-level deep learning library.
d) none of the above

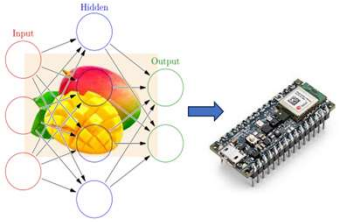
What is Keras and Pandas?. . .



Why Keras Is Popular?

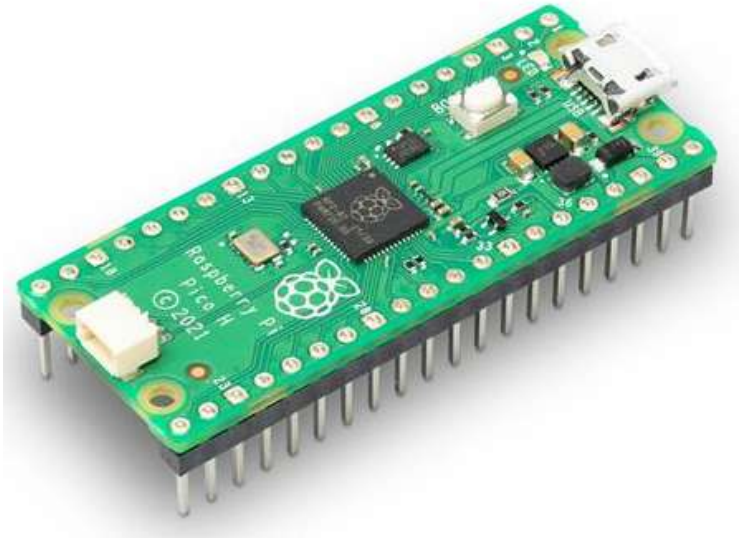
- **Simple and intuitive**
  You can build a neural network in 5–10 lines of code.
- **Runs on the TensorFlow engine**
  You get all the power of TensorFlow (GPU acceleration, efficient computation) with a clean, easy interface.
- **Supports many model types**
  a) Dense (fully connected) neural networks
  b) Convolutional Neural Networks (CNNs)
  c) Recurrent Neural Networks (RNNs)
  d) Transformers
  e) Autoencoders
  f) Long Short-Term Memory (LSTM) and Gate Recurrent Unit (GRU) networks
- **Works with real-world data**
  Images, audio, CSV files, text, sensor data … all supported.
  Works on big systems and small ones (TinyML)

What is Keras and Pandas?. . .

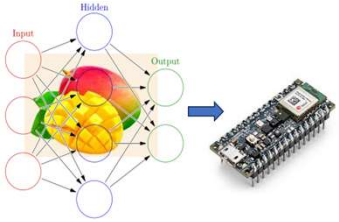


What is Keras?

Raspberry Pi Pico



What is Keras and Pandas?. . .

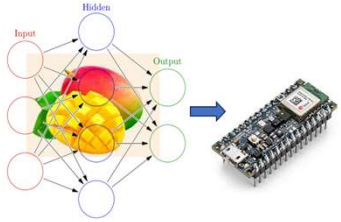


Summary

Keras is:

- A high-level, beginner-friendly deep learning library that lets you easily build and train neural networks.
- It sits on top of TensorFlow and allows you to move from
  a) data → trained model
  b) trained model → deployment with very little overhead

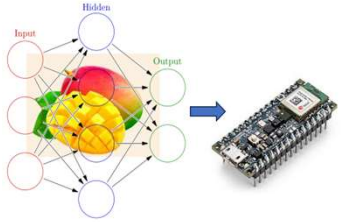
What is Keras and Pandas?. . .



What is Pandas?

- Pandas is a powerful, easy-to-use Python library for working with data, especially data in tables (rows and columns), much like an Excel spreadsheet.
- It is one of the most important tools in data science, machine learning, and engineering analytics.

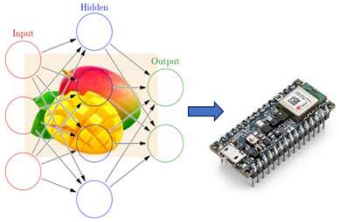
What is Keras and Pandas?. . .



What is Pandas?

- **Pandas** is a powerful, easy-to-use Python library for working with data, especially data in tables (rows and columns), much like an Excel spreadsheet.
- It is one of the most important tools in data science, machine learning, and engineering analytics.

What is Keras and Pandas?. . .

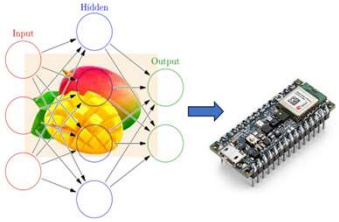


What Pandas Does?

Pandas helps you:

a) Load data (CSV, Excel, JSON, SQL, sensor logs, etc.)
b) Organize and clean data
c) Filter, sort, and transform data
d) Analyze data quickly
e) Prepare data for machine learning models

What is Keras and Pandas?. . .



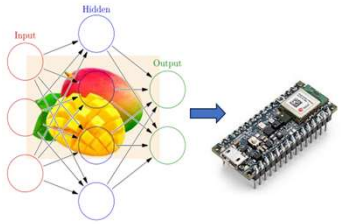
Core Data Structures

Pandas provides two main objects:

a. Series
- A single column of data
- Like a list with labels

b. DataFrame
- A full table with rows and columns
- Like an Excel sheet inside Python
- This is what you'll use 90% of the time

What is Keras and Pandas?. . .



Example Data Frame:

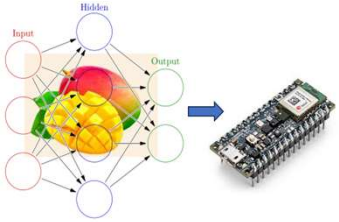
R	G	B	fruit_label
215	34	30	apple
250	220	15	banana
200	115	10	orange

Question 4

What does slide 29 represent?

a) A neural network
b) Pandas Data Frame
c) Keras framework
d) none of the above

What is Keras and Pandas?. . .

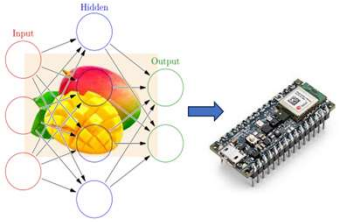


Why Pandas Matters for Machine Learning?
Before training a model in Keras, TensorFlow, or scikit-learn, you often must:

a) clean messy data
b) normalize or scale values
c) split data into training/testing
e) handle missing values
d) convert labels to numbers

Pandas do all of that easily.

What is Keras and Pandas?. . .

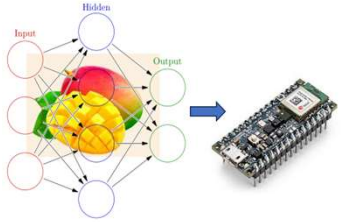


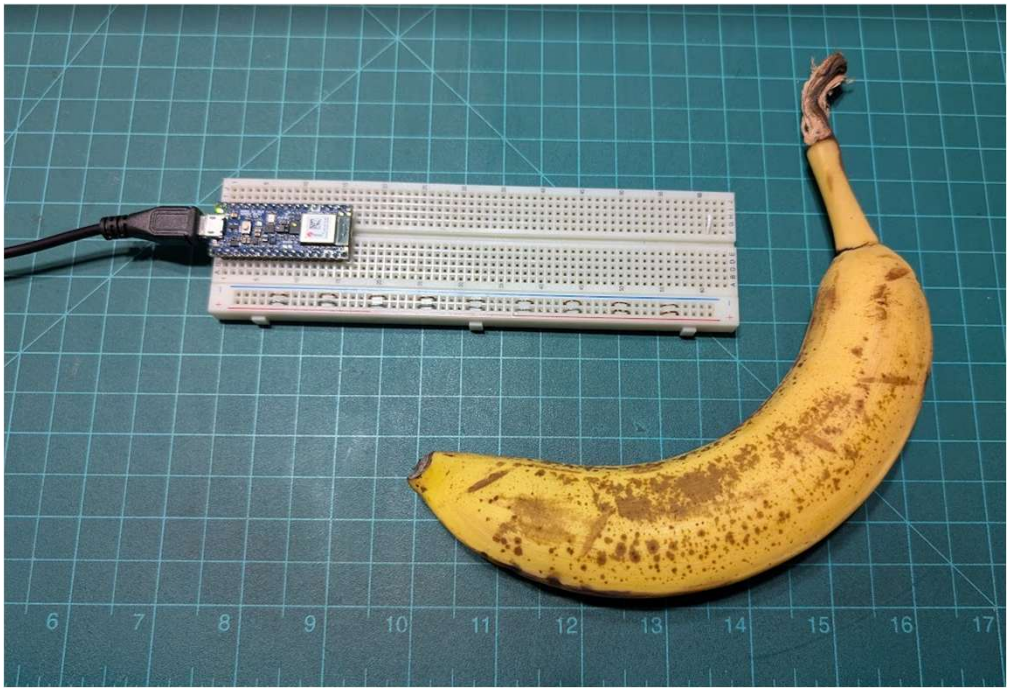
Summary

- **Pandas** is a Python data analysis library that gives you powerful, flexible tools to work with data in a table format.
- Performing any machine learning event
  a) sensor analysis
  b) robotics logging,
  c) or engineering analytics,

Pandas is essential.

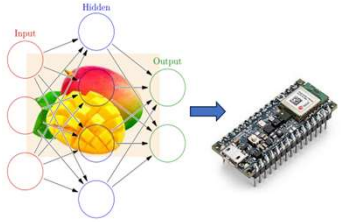
Lab: Building a Fruit To Emoji Classifier





```
=== Classification Results ===
       Red  Green   Blue          Fruit Predicted_Fruit
0    0.437  0.310  0.254  banana (~0~)    banana (~0~)
1    0.436  0.310  0.254  banana (~0~)    banana (~0~)
2    0.437  0.310  0.253  banana (~0~)    banana (~0~)
3    0.436  0.310  0.254  banana (~0~)    banana (~0~)
4    0.438  0.309  0.253  banana (~0~)    banana (~0~)
..     ...    ...    ...           ...             ...
488  0.462  0.308  0.231       unknown         unknown
489  0.462  0.308  0.231       unknown         unknown
490  0.462  0.308  0.231       unknown         unknown
491  0.462  0.308  0.231       unknown         unknown
492  0.455  0.364  0.182      orange 0         unknown

[493 rows x 5 columns]
```

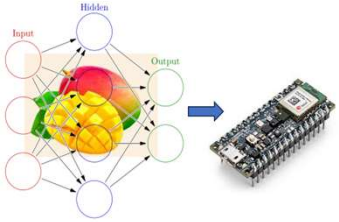
Lab: Building a Fruit To Emoji Classifier. . .



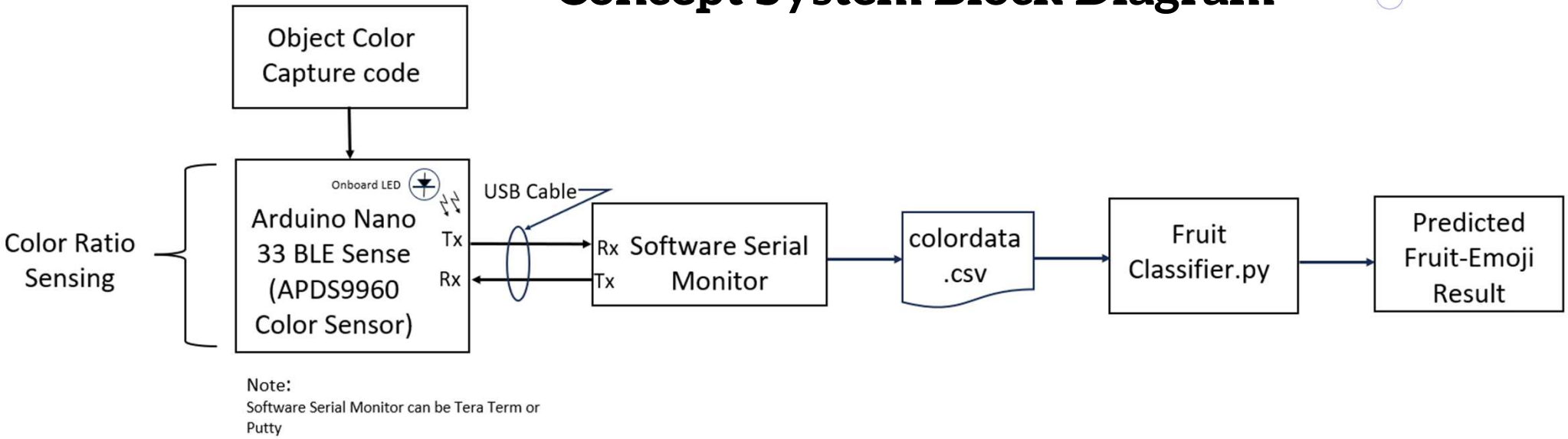
Lab Objectives:

- Participants will learn to set up a Fruit Classifier System using the Arduino Nano 33 BLE Sense board, color capture software, and classification Python code.
- Participants will learn to upload the color capture software to the Arduino Nano 33 BLE Sense board.
- Participants will learn to perform a color ratio data collection session.
- Participants will learn to use a classifier Python code for fruit recognition with the appropriate emoji

Lab: Building a Fruit To Emoji Classifier. . .

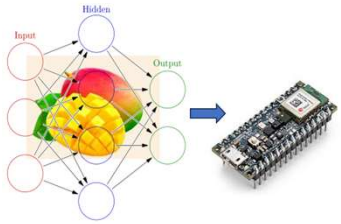


Concept System Block Diagram



Setup-Fruit Classifier System

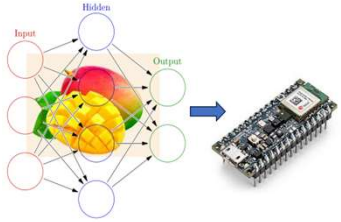
Lab: Building a Fruit To Emoji Classifier. . .




```
#include <Arduino_APDS9960.h>

#include <TensorFlowLite.h>
#include <tensorflow/lite/micro/all_ops_resolver.h>
#include <tensorflow/lite/micro/micro_interpreter.h>
#include <tensorflow/lite/schema/schema_generated.h>

#include "model.h"   // provides model_tflite[] and model_tflite_len

// TFLite objects
const tflite::Model* tflModel = nullptr;
tflite::MicroInterpreter* tflInterpreter = nullptr;
TfLiteTensor* inputTensor = nullptr;
TfLiteTensor* outputTensor = nullptr;

constexpr int kTensorArenaSize = 16 * 1024;
uint8_t tensor_arena[kTensorArenaSize];

tflite::AllOpsResolver resolver;

// Class labels
const char* CLASSES[] = { "Apple", "Banana", "Orange" };
#define NUM_CLASSES 3

void setup() {
  Serial.begin(115200);
  while (!Serial) {}
```

Partial Object Color Capture Code

Lab: Building a Fruit To Emoji Classifier. . .



Color Data streaming from the APDS9960 Sensor

COM12 - Tera Term VT

File Edit Setup Control Window Help

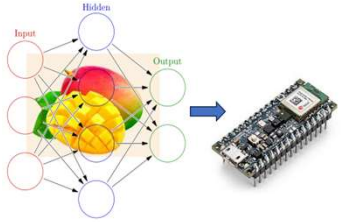
```
Red,Green,Blue
0.416,0.323,0.261
0.416,0.323,0.262
0.415,0.324,0.261
0.415,0.324,0.261
0.416,0.323,0.260
0.415,0.324,0.261
0.415,0.324,0.261
0.416,0.323,0.260
0.416,0.324,0.261
0.416,0.323,0.260
0.415,0.324,0.261
0.416,0.323,0.262
0.415,0.324,0.261
0.416,0.323,0.260
0.414,0.325,0.261
0.416,0.323,0.260
0.415,0.324,0.261
0.416,0.323,0.261
0.416,0.323,0.260
0.416,0.323,0.261
0.416,0.323,0.260
0.414,0.325,0.261
0.415,0.324,0.261
```

Software Serial Monitor Logging Session

- Color Data streaming from the APDS9960 sensor
- Start a data logging session: File>Log>folder selected>name.csv
- Click the save button

Lab: Building a Fruit To Emoji Classifier. . .

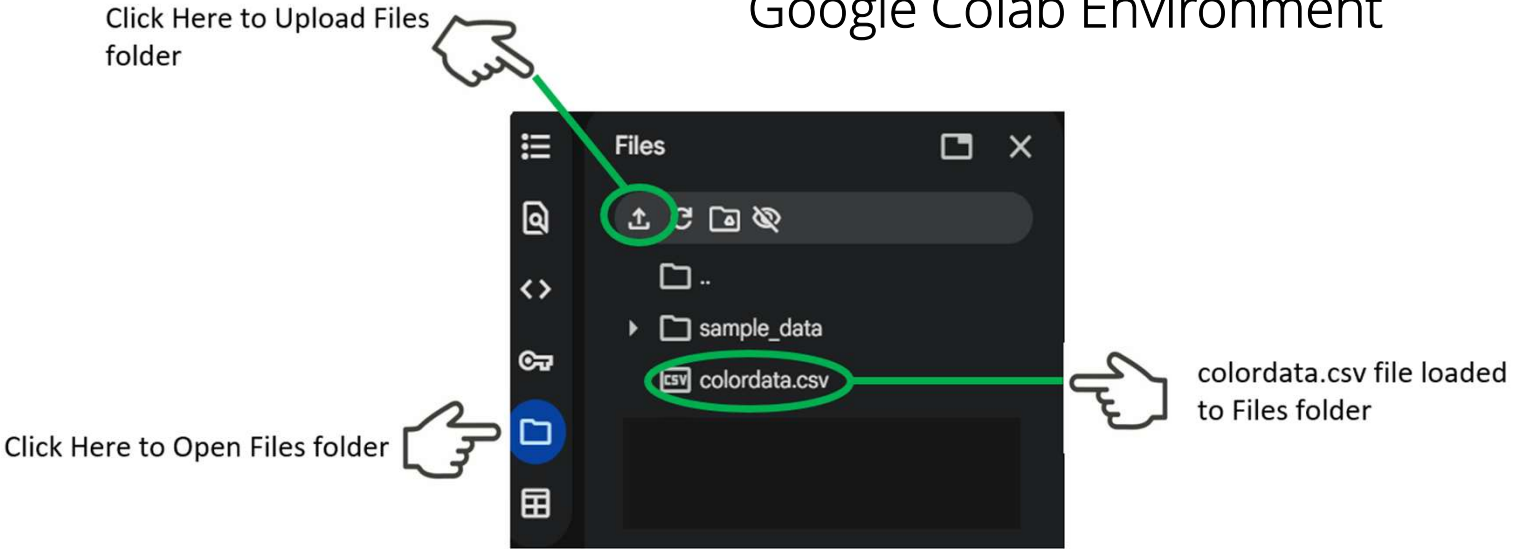
Uploading .csv file to the Google Colab Environment



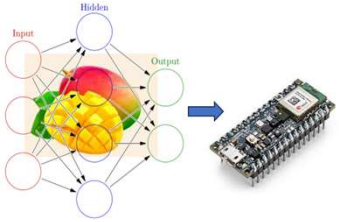
Partial colordata.csv file

	A	B	C
1	Red	Green	Blue
2	0.437	0.31	0.254
3	0.436	0.31	0.254
4	0.437	0.31	0.253
5	0.436	0.31	0.254
6	0.438	0.309	0.253
7	0.429	0.308	0.264
8	0.43	0.301	0.269
9	0.445	0.299	0.256
10	0.449	0.298	0.253
11	0.448	0.3	0.251
12	0.447	0.299	0.254
13	0.45	0.3	0.25
14	0.449	0.301	0.25
15	0.442	0.303	0.255
16	0.437	0.297	0.266
17	0.43	0.301	0.269
18	0.431	0.304	0.265

Google Colab Environment



Lab: Building a Fruit To Emoji Classifier. . .

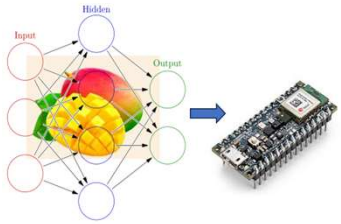


Execute Fruit_Classifier.py (Parts A and B) in Google Colab Environment

Part A Partial Code Listing

```
[1]
✓ 0s
1   import pandas as pd
2
3   # ---------------------------------------------------------
4   # Load the CSV file generated by your Arduino color sampler
5   # ---------------------------------------------------------
6   csv_file = "colordata.csv"   # <-- change to your file name
7   df = pd.read_csv(csv_file, header=None)
8   df.columns = ['Red', 'Green', 'Blue']
9
10  print("Loaded Data:")
11  print(df.head())
12
13  # ---------------------------------------------------------
14  # Fruit Classification Rules
15  # ---------------------------------------------------------
16  def classify_fruit(r, g, b):
17      # These are approximate color characteristics:
18      # Apple: higher red
19      # Banana: higher green (yellow = red+green, but green dominates)
20      # Orange: red moderately high, green moderate, blue low
21
22      if r > 0.50 and g < 0.35:
23          return "apple (o)"
24
25      #elif g > r and g <0.45 and b >.250 :
26      elif r >.4 and g < 0.45 and b > .250:
```

Lab: Building a Fruit To Emoji Classifier. . .

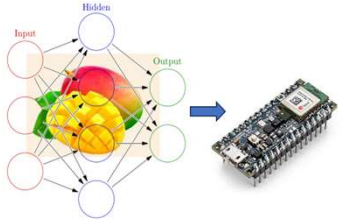


Execute Fruit_Classifier.py (Parts A and B) in Google Colab Environment. . .

Part B Partial Code Listing

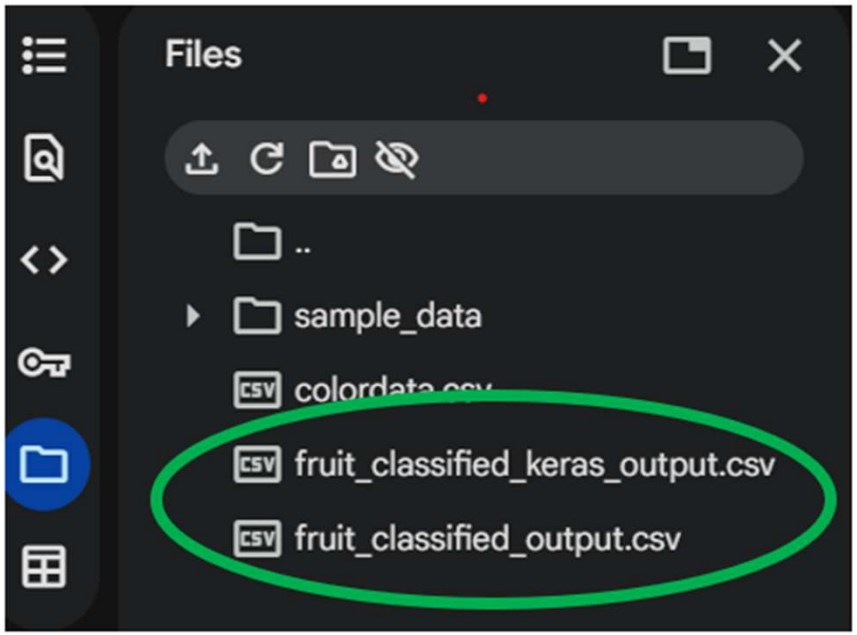
```python
import pandas as pd
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

# ------------------------------------------------------------
# Load the CSV file containing the classified fruit data
# ------------------------------------------------------------
csv_file = "fruit_classified_output.csv"   # <-- Load the output from the previous classification step
df = pd.read_csv(csv_file)

print("Loaded Data:")
print(df.head())

# ------------------------------------------------------------
# Prepare features (RGB) and labels (Fruit)
# ------------------------------------------------------------
X = df[["Red", "Green", "Blue"]].values  # features

# Encode fruit labels to integers
label_encoder = LabelEncoder()
```

Lab: Building a Fruit To Emoji Classifier. . .



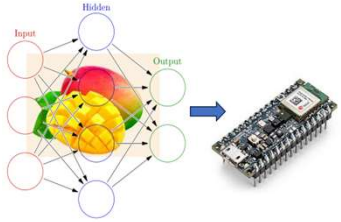
Execute Fruit_Classifier.py (Parts A and B) in Google Colab Environment. . .

Two Files are generated after Part B Fruit Classifier is executed



Generated .csv files

Lab: Building a Fruit To Emoji Classifier. . .



Execute Fruit_Classifier.py (Parts A and B) in Google Colab Environment. . .

fruit_classified_keras_output.csv fru

1 to 10 of 493 entries Filter

Red	Green	Blue	Fruit	Predicted_Fruit
0.437	0.31	0.254	banana (~0~)	banana (~0~)
0.436	0.31	0.254	banana (~0~)	banana (~0~)
0.437	0.31	0.253	banana (~0~)	banana (~0~)
0.436	0.31	0.254	banana (~0~)	banana (~0~)
0.438	0.309	0.253	banana (~0~)	banana (~0~)
0.429	0.308	0.264	banana (~0~)	banana (~0~)
0.43	0.301	0.269	banana (~0~)	banana (~0~)
0.445	0.299	0.256	banana (~0~)	unknown
0.449	0.298	0.253	banana (~0~)	unknown
0.448	0.3	0.251	banana (~0~)	unknown

Show 10 per page 1 2 10 40 50

ıt.csv fruit_classified_output.csv

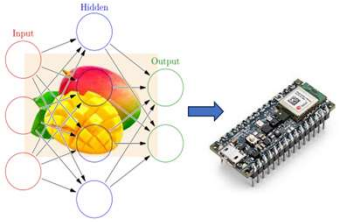
1 to 10 of 493 entries Filter

Red	Green	Blue	Fruit
0.437	0.31	0.254	banana (~0~)
0.436	0.31	0.254	banana (~0~)
0.437	0.31	0.253	banana (~0~)
0.436	0.31	0.254	banana (~0~)
0.438	0.309	0.253	banana (~0~)
0.429	0.308	0.264	banana (~0~)
0.43	0.301	0.269	banana (~0~)
0.445	0.299	0.256	banana (~0~)
0.449	0.298	0.253	banana (~0~)
0.448	0.3	0.251	banana (~0~)

Show 10 per page 1 2 10 40 50

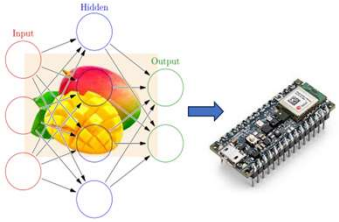
Results from the two generated files

Lab: Building a Fruit To Emoji Classifier. . .

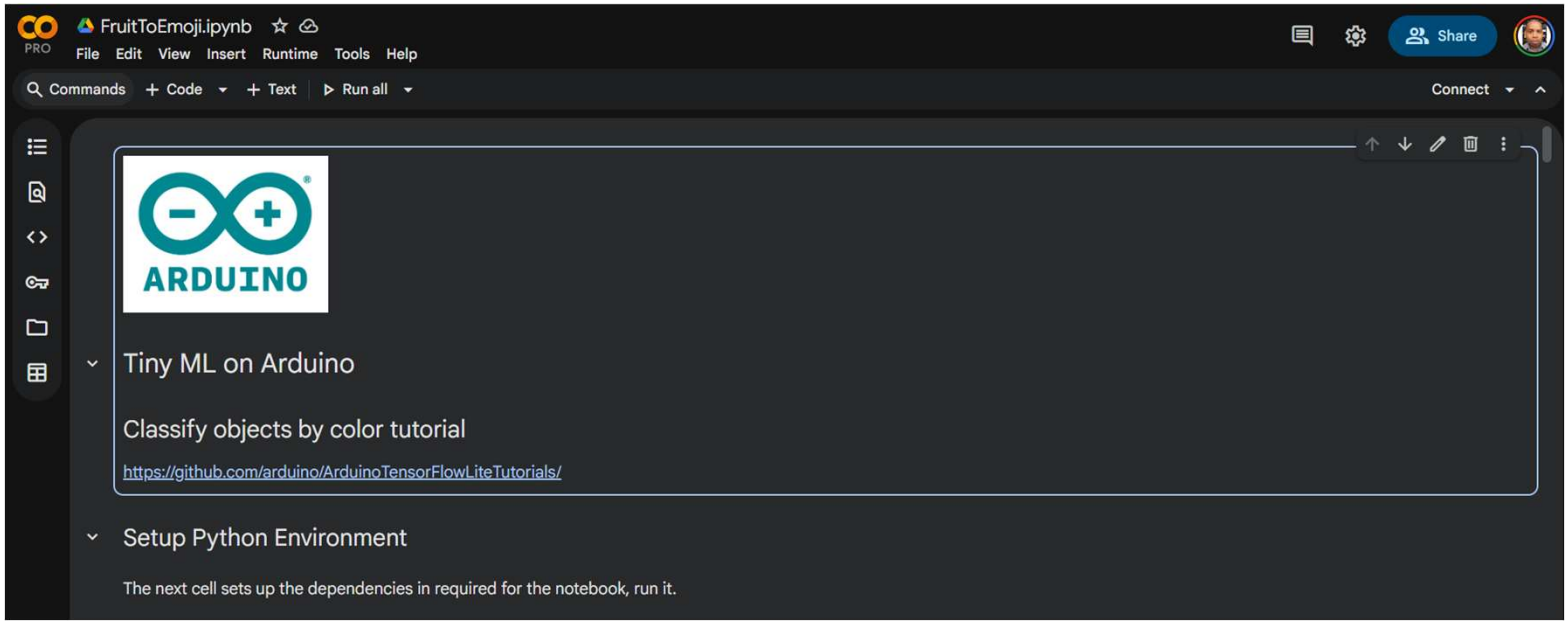


Open this File to Train for other fruit or object recognition!

Lab: Building a Fruit To Emoji Classifier. . .



Open this File to Train for other fruit or object recognition!



FruitToEmoji.ipynb (Google Colab Notebook)

Question 5

In reviewing slide 36, which line number has the TensorFlow Lite-trained model for a specific fruit?

a) 1
b) 13
c) 8
d) none of the above

Thank you for attending

Please consider the resources below:

[1] J. Lin, L. Zhu, W. M. Chen, W. C.Wang, and S. Han, "Tiny machine learning: Progress and futures," *arXiv*:2403.19076v2 [cs.LG], Jun. 2016. [Online]. Available: https://arxiv.org/abs/2403.19076
[2] R. Mathur, "A detailed intro to neural networks," Aug. 2023. [Online]. Available: https://rikinmathur.substack.com/p/a-detailed-intro-to-neural-networks
[3] S. Heydari, Q. H. Mahmoud, "Tiny machine learning and on-device inference: A survey of applications, challenges, and future directions," May. 2025. [Online]. Available: https://www.mdpi.com/1424-8220/25/10/3191
[4] D. Wilcher, "Designs News December 25 webinar code," GitHub repository, Dec. 2025. [Online]. Available: https://github.com/DWilcher/DesignNews-WebinarCode/blob/main/December_25_Webinar_Code.zip
[5] A. Dehghani, O. Sarbishei, T. Glatard, and E. Shihab, " A quantitative comparison of overlapping and non-overlapping sliding windows for human activity recognition using inertial sensors," Nov. 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/22/5026

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

DigiKey

informa markets