

TinyMLjs

Training an ML
microcontroller model
using a static webpage



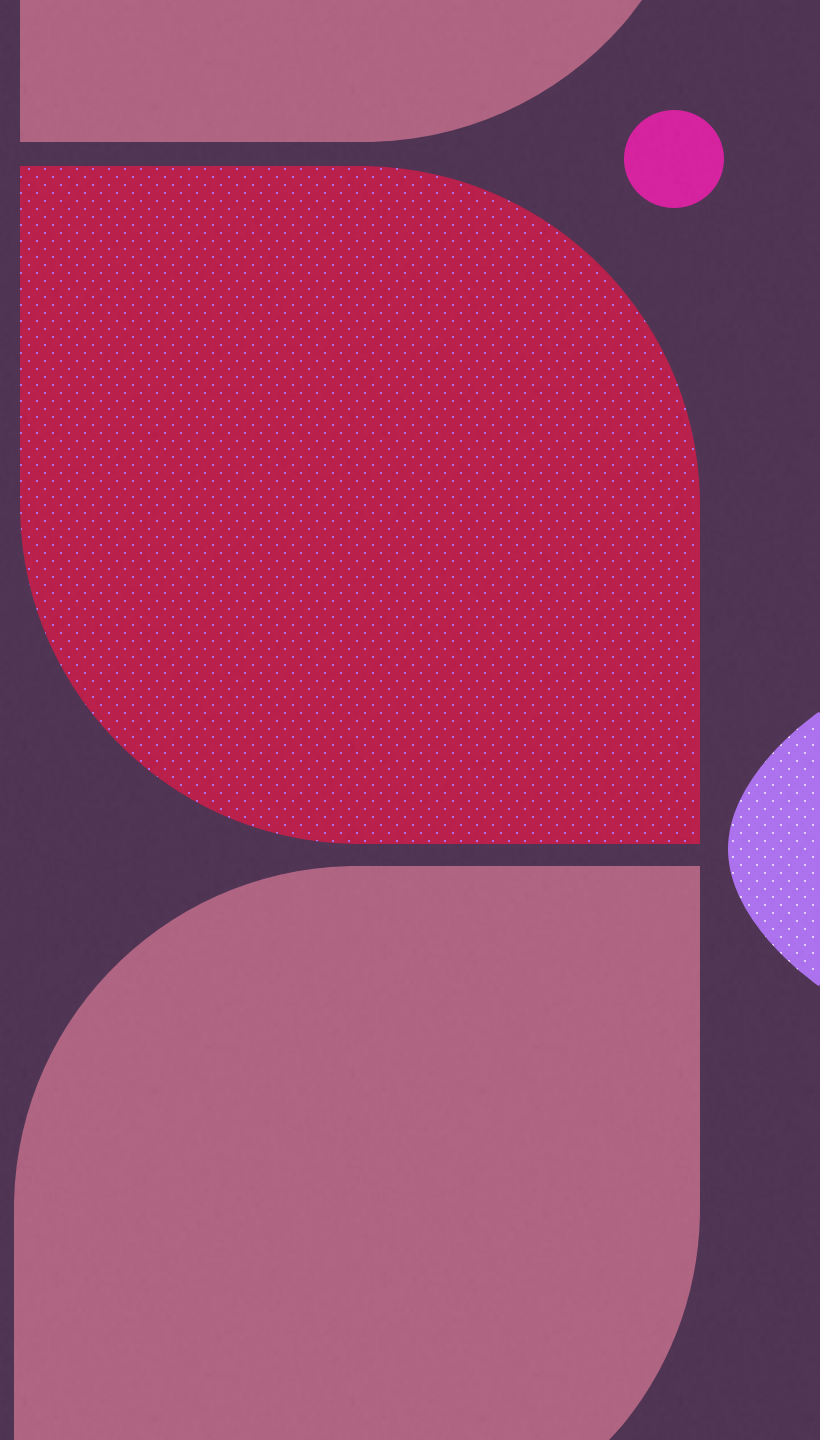
Demo on Cell phone or laptop

<https://hpssjellis.github.io/tinyMLjs/public/acceleration/a00-best-acceleration-rak2270-sticker-tracker.html>



Why use Javascript to train ML for microcontrollers?

- 1. Reduce deprecation issues
- 2. Reduce cloud costs
- 3. Full control of everything
- 4. True data privacy
- 5. No need for an internet connection
- 6. Huge WebML community
- 7. Ease of use for beginners and students



Start the SPA (single page application)

TinyMLjs

Making TinyML truly client-side. Giving Makers full control of the process, user friendly, private and protected

Version 0.52.3-211 Note: WebSerial microcontroller connection works on Chrome or Edge for Mac, Android (pixel Phones) or Windows, only works on Edge for Linux

This website makes machine learning models from WebSerial connected micro-controller sensors using TensorflowJS. That model can be saved and converted to a tFlite model then to a C-header model.h file using Tensorflow command line or Python converters. I simplify the conversion using a Gitpod. Finally the model.h file is combined with C/C++ code using an Arduino IDE ready library called RocksettaTinyML and compiled to the device for testing.

Presently for complex vision or sound data it is easier to use [EdgeImpulse.com](#) as it will achieve the needed model compression that we have not yet achieved.

Connect via Serial Port

start

send 'a' LED On or Off

Clear and send 'start'

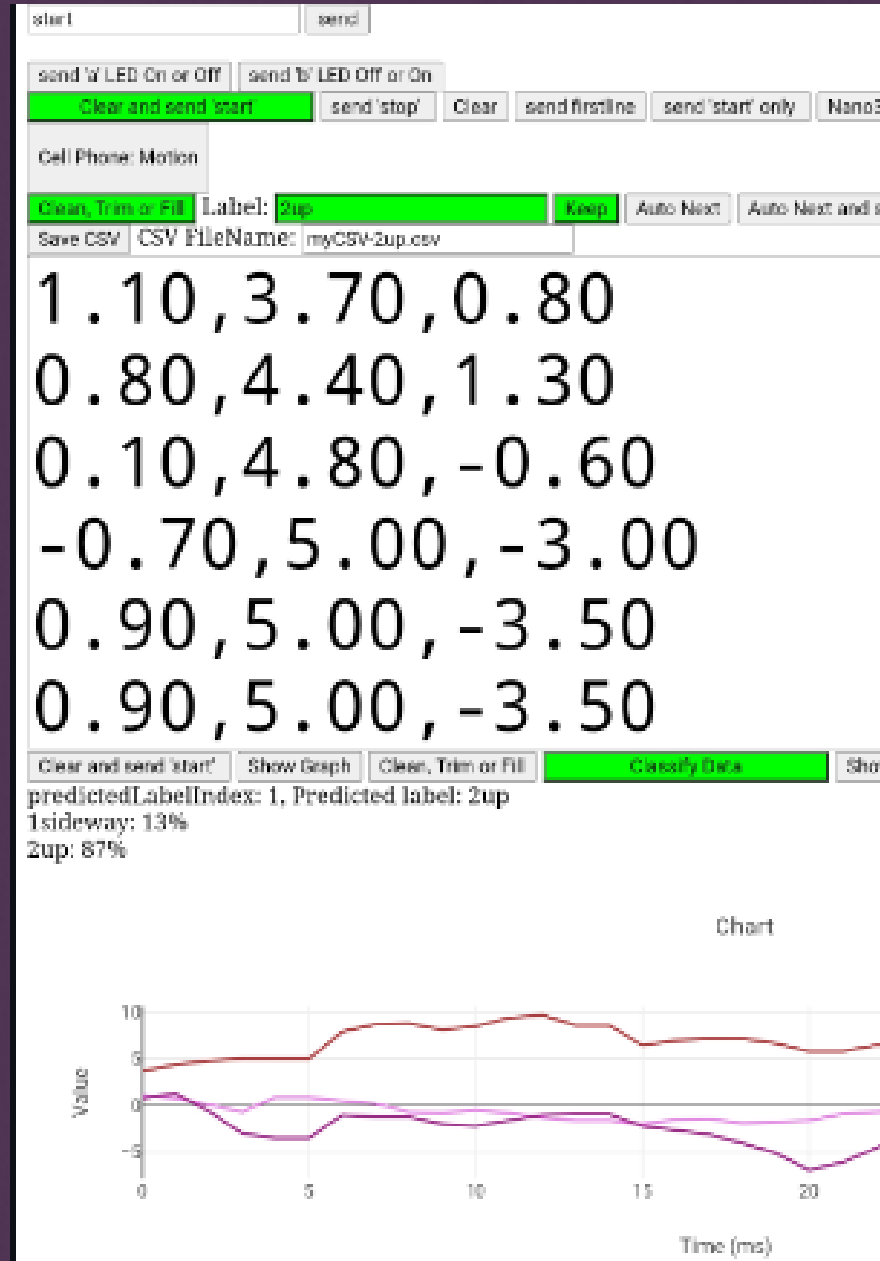
Cell Phone: Motion

Clean, Trim or Fill

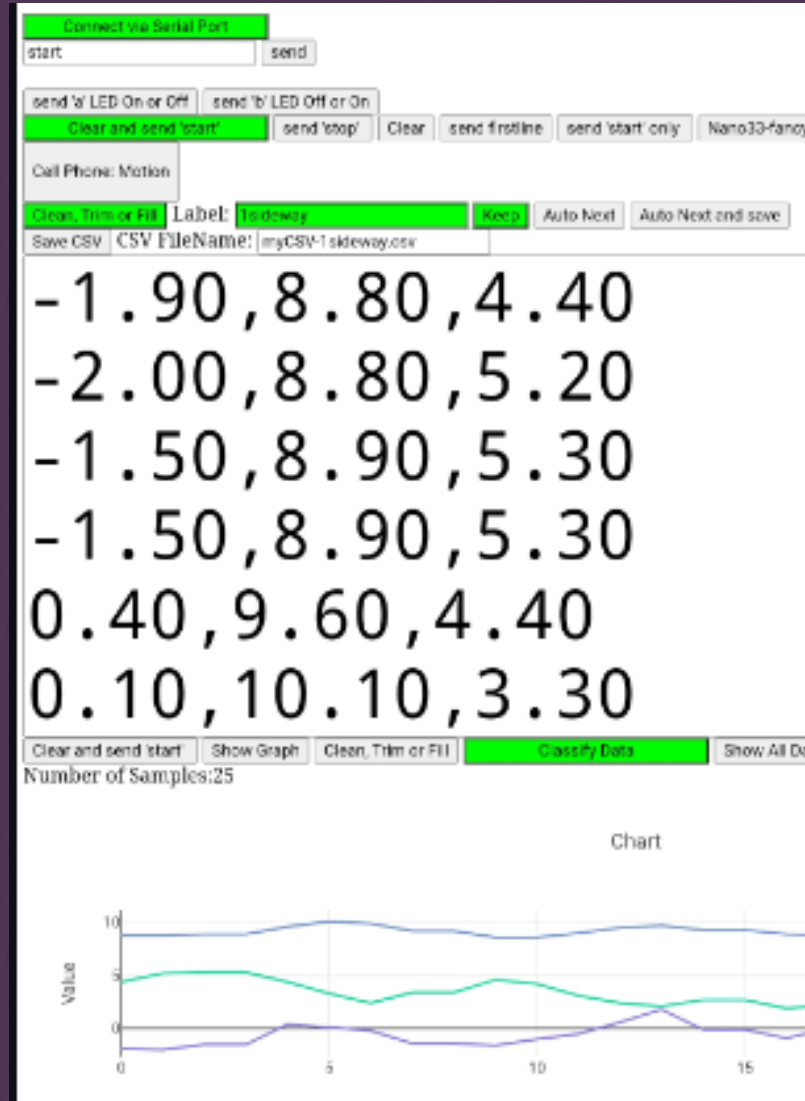
Save CSV

...

Collect Data



I do 3 moving sideways and 3 moving up. Change the label!



Clean, Trim and Fill, then
convert to tensor, then train

[Click here to see the working HTML code.](#)

Convert Data to Tensor

Enter number of epochs: , Learning rate:

Epoch 15: loss =
0.510303497314453
= 0.833333337306970

Train Model

Just Fit - retrain

View Model

Possible to both save and upload saved CSV file data

Hide File Uploading

Upload from a raw CSV file or an Arduino style microcontroller using webSerial (Android Pixel phones also work) or your cell phone motion sensor. Keep the raw sensor data then Machine Learning train a tensorflowJS model for export or for live classification all on this single vanilla Javascript webpage!

Show: ☒ ----- Hide: ☐ ----- Load .csv files: No file chosen

Following is the list of actual labels used in the same order as uploaded (comma-separated) Note: expecting files to be named: "name-label.csv" or "name-label (1).csv" etc.

CSV Lables (careful):

Senses Labels (In the order collected):

Select model file (.json): CoolMod.json

Select weights file (.bin): CoolMod.weights.bin

Select labels file (.txt): CoolModLabels.txt

Enter Labels (comma seperated):

Now the hard part, converting tfjs to tflite then a c-header file

Train Model

Just Fit - retrain

View Model

Layer 1:flatten_Flatten3, outShape ,108, params: 0, trainable: true
Layer 2:dense_Dense9, outShape ,8, params: 872, trainable: true
Layer 3:dense_Dense10, outShape ,20, params: 180, trainable: true
Layer 4:dense_Dense11, outShape ,30, params: 630, trainable: true
Layer 5:dense_Dense12, outShape ,2, params: 62, trainable: true

Export Model

model

Try the following steps:

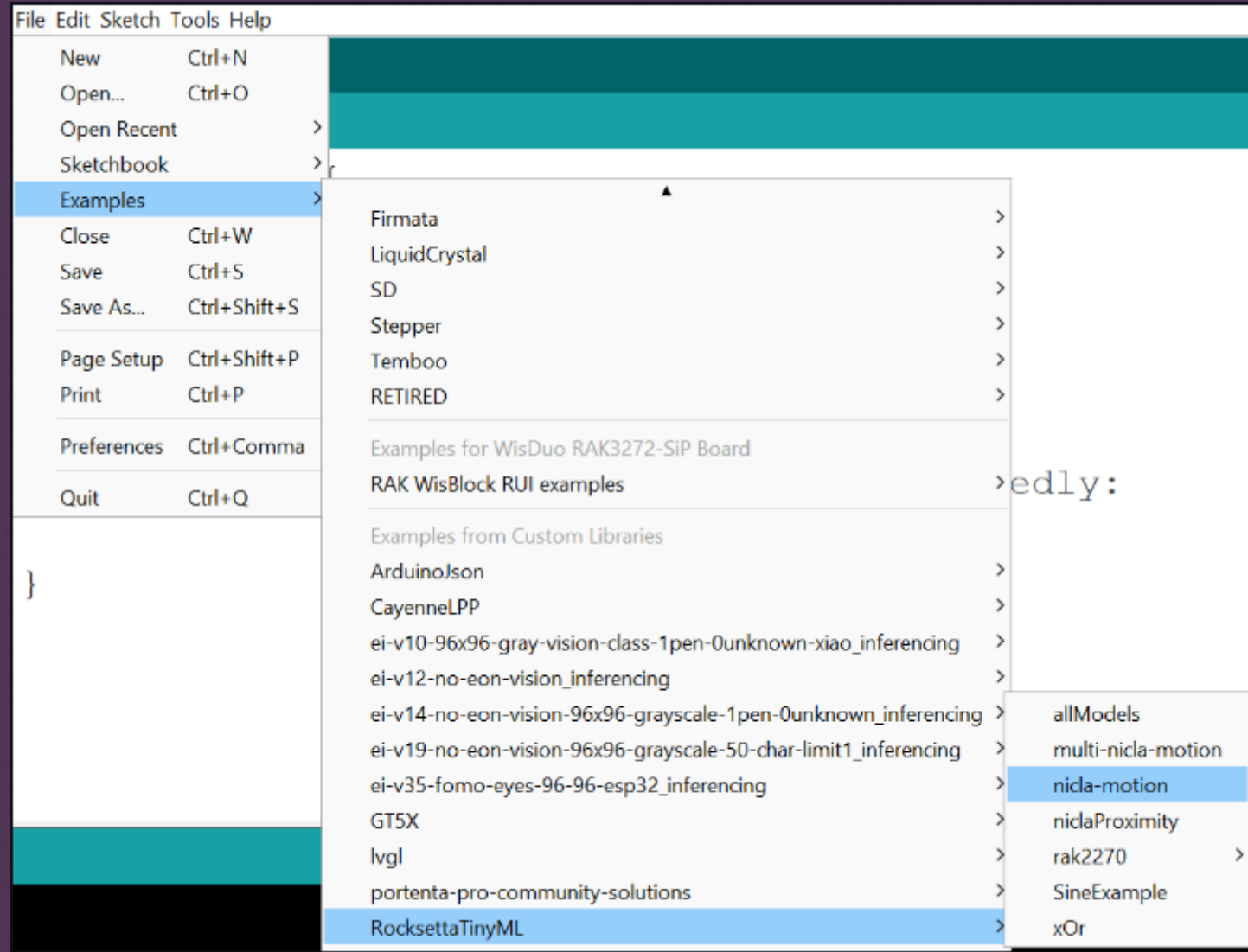
1. Convert the exported model to Arduino ready c-header model.h file as well as a model.tflite file using 1 of the following methods:

1:	tensorflowjs-to-arduino-for-tinymlops	Best to install the modules needed to client side do the conversions yourself. This github shows what you need to do. The installation might be different on your computer.
2:	iPython Notebook TFJS to TFlite	This web based Google Colab iPython notebook (Needs a google login) loads the necessary code then you click on an upload button to load your tensorflowjs exported "model.json" and "model.weights.bin" files and it zips and downloads the tflite and c-header files.
3:	Use a Gitpod: tensorflowjs-to-arduino-for-tinymlops or direct load: Gitpod	A gitpod browser docker like program that auto loads the necessary python files and then runs a bash program to do the command line conversions. All code is easy to view

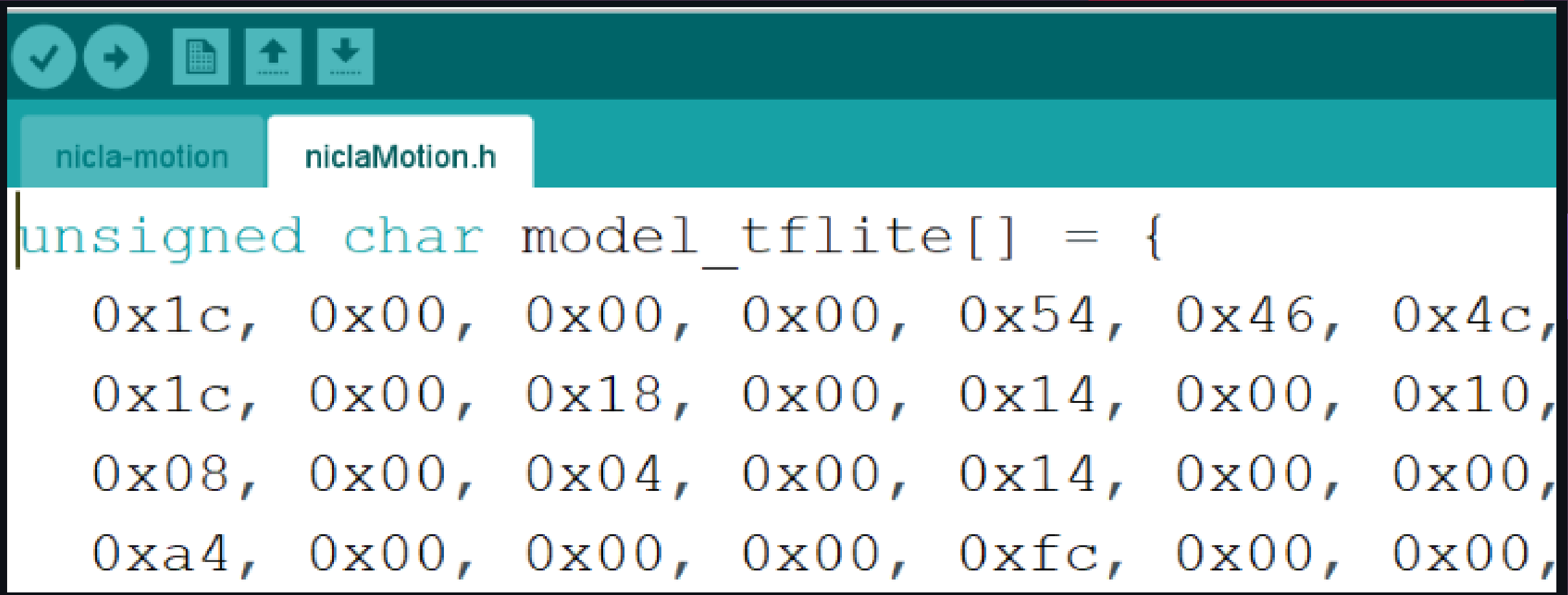
3. Use <https://netron.app/> Use the online netron.app to check and visualize your downloaded model.tflite
4. Once you have made a model.h file then install this Arduino Library [RocksettaTinyML](#), based on EloquentArduino to load the code onto your Arduino IDE.

...

Once RocksettaTinyML installed on the Arduino IDE, load an example and...



Replace the c-header file with your new c-header file

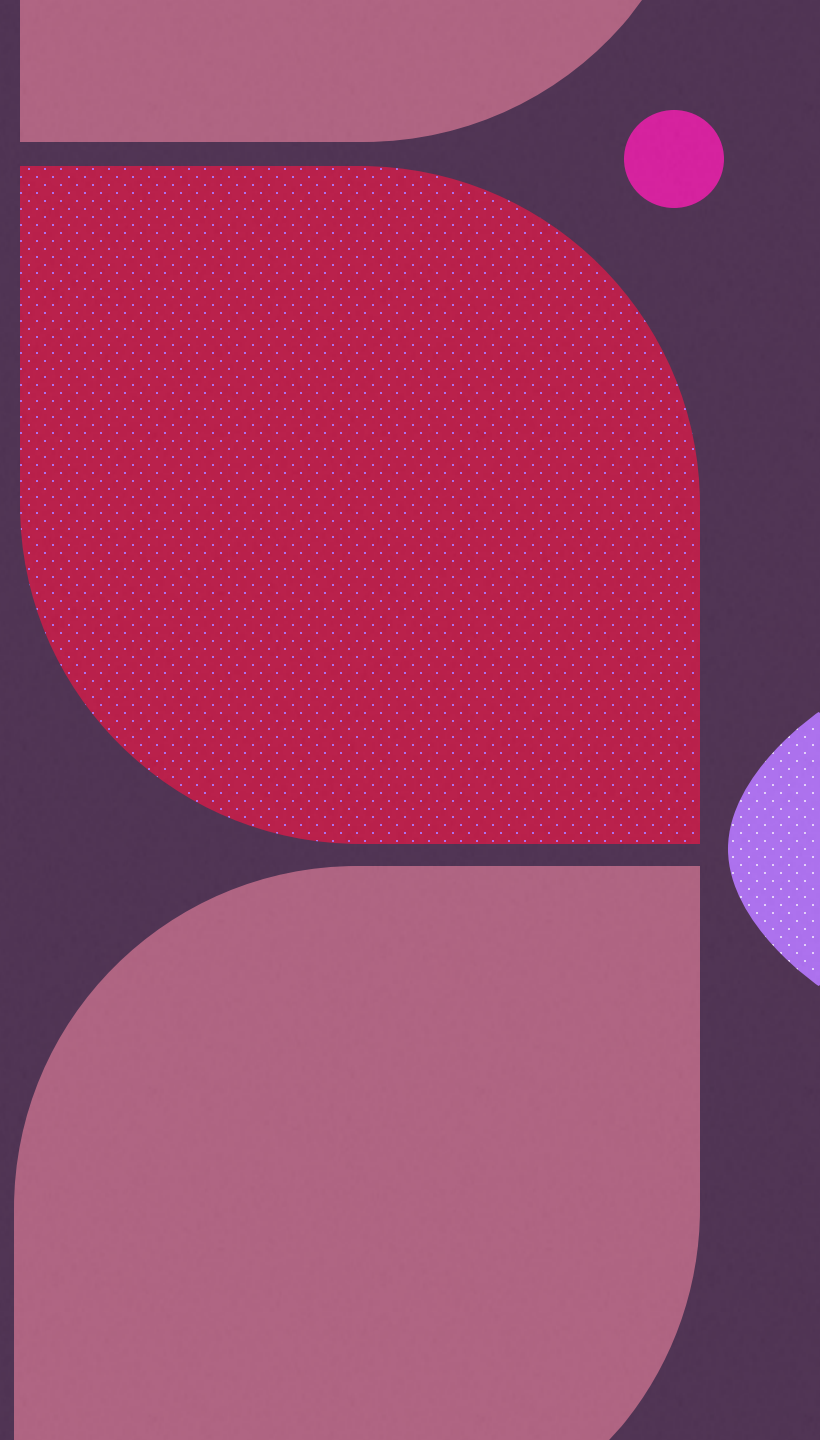


The screenshot shows a code editor interface with a teal header bar. On the left, there are five icons: a checkmark, a right arrow, a document, an up arrow, and a down arrow. Below the header bar, there are two tabs: 'nicla-motion' and 'niclaMotion.h'. The 'niclaMotion.h' tab is active, displaying the following C code:

```
unsigned char model_tfllite[] = {  
    0x1c, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c,  
    0x1c, 0x00, 0x18, 0x00, 0x14, 0x00, 0x10,  
    0x08, 0x00, 0x04, 0x00, 0x14, 0x00, 0x00,  
    0xa4, 0x00, 0x00, 0x00, 0xfc, 0x00, 0x00,
```

Why use Javascript to train ML for microcontrollers?

- 1. Reduce deprecation issues
- 2. Reduce cloud costs
- 3. Full control of everything
- 4. True data privacy
- 5. No need for an internet connection
- 6. Huge WebML community
- 7. Ease of use for beginners and students



TinyMLjs Demo

- A new possibility for Educators teaching Machine Learning for Microcontrollers