





## 269 lines - 26 Removals

- 1 /\* Edge Impulse Arduino examples
- 2 \* Copyright (c) 2021 EdgeImpulse Inc.
- o 8
- \* Permission is hereby granted, free of charge, to any person obtaining a copy
- 5 \* of this software and associated documentation files
   (the "Software"), to deal
- \* in the Software without restriction, including witho ut limitation the rights
- 7 \* to use, copy, modify, merge, publish, distribute, su blicense, and/or sell
- 8 \* copies of the Software, and to permit persons to who
  m the Software is
- 9 \* furnished to do so, subject to the following conditions:
- 10 \*
- 11 \* The above copyright notice and this permission notic e shall be included in
- \* all copies or substantial portions of the Software.
- 13
- 14 \* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY O F ANY KIND, EXPRESS OR
- 15 \* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
  OF MERCHANTABILITY,
- 16 \* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMEN
  T. IN NO EVENT SHALL THE
- 17 \* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAI M, DAMAGES OR OTHER
- 18 \* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
- 19 \* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
- 20 \* SOFTWARE.
- 21 \*/

22

- 23 // If your target is limited in memory remove this macr o to save 10K RAM
- 24 #define EIDSP\_QUANTIZE\_FILTERBANK 0
- 26 /\*:
- 27 \* Define the number of slices per model window. E.g. a model window of 1000 ms
- 28 \* with slices per model window set to 4. Results in a slice size of 250 ms.
- 29 \* For more info: https://docs.edgeimpulse.com/docs/con tinuous-audio-sampling
- 30 \*/
- 31 //#define EI\_CLASSIFIER\_SLICES\_PER\_MODEL\_WINDOW 3
- 32 33 /\*
- $^{34}$   $\ \ **$  NOTE: If you run into TFLite arena allocation issu

## 330 lines + 89 Additions

- 3 /\* Edge Impulse Arduino examples
- 4 \* Copyright (c) 2021 EdgeImpulse Inc.
- \_ x
- \* Permission is hereby granted, free of charge, to any
  person obtaining a copy
- 7 \* of this software and associated documentation files
   (the "Software"), to deal
- $8\ \ *$  in the Software without restriction, including witho ut limitation the rights
- 9 \* to use, copy, modify, merge, publish, distribute, su blicense, and/or sell
- 10  $\,^*$  copies of the Software, and to permit persons to who  $\,^{\mathrm{m}}$  the Software is
- 11 \* furnished to do so, subject to the following conditi
  ons:
- 12 \*
- 13 \* The above copyright notice and this permission notic e shall be included in
- \* all copies or substantial portions of the Software.
- 15 \*
- 16 \* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY O F ANY KIND, EXPRESS OR
- 17 \* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
  OF MERCHANTABILITY,
- 18 \* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMEN
  - T. IN NO EVENT SHALL THE
- 19 \* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAI M, DAMAGES OR OTHER
- 20 \* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
- 21 \* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
- 22 \* SOFTWARE.
- 23 \*

25

- 24 #include "RPC.h" // comes with the mbed board installa
  tion
- 26 // If your target is limited in memory remove this macr o to save 10K RAM
- 27 #define EIDSP\_QUANTIZE\_FILTERBANK 0
- 28 29 /\*\*
- 29 /\*
- $^{\ast}$  Define the number of slices per model window. E.g. a model window of 1000 ms
- 31  $\,\,$  \* with slices per model window set to 4. Results in a slice size of 250 ms.
- 32 \* For more info: https://docs.edgeimpulse.com/docs/con tinuous-audio-sampling
- 33 \*
- 34 //#define EI\_CLASSIFIER\_SLICES\_PER\_MODEL\_WINDOW 3
- 35 36 /\*
- $^{\rm 37}$   $\,$  \*\* NOTE: If you run into TFLite arena allocation issu

```
e.
35 **
   ** This may be due to may dynamic memory fragmentatio
37 ** Try defining "-DEI_CLASSIFIER_ALLOCATION_STATIC" in
   boards.local.txt (create
38 ** if it doesn't exist) and copy this file to
39 ** `<ARDUINO_CORE_INSTALL_PATH>/arduino/hardware/<mbed</pre>
   _core>/<core_version>/`.
40 **
41 ** See
42 ** (https://support.arduino.cc/hc/en-us/articles/36001
   2076960-Where-are-the-installed-cores-located-)
** to find where Arduino installs cores on your machin
   e.
   ** If the problem persists then there's not enough mem
   ory for this model and application.
46 */
47
48 /* Includes -----
   */
49 #include <PDM.h>
50 #include <FYP Direction inferencing.h>
51
52 /** Audio buffers, pointers and selectors */
53 typedef struct {
       signed short *buffers[2];
       unsigned char buf select;
55
      unsigned char buf_ready;
57
      unsigned int buf count;
       unsigned int n_samples;
59 } inference_t;
61 static inference_t inference;
62 static volatile bool record ready = false;
63 // static signed short *sampleBuffer;
64 static signed short sampleBuffer[2048];
65 static bool debug_nn = false; // Set this to true to se
   e e.g. features generated from the raw signal
66 static int print_results = -(EI_CLASSIFIER_SLICES_PER_M
   ODEL_WINDOW);
67
68 /**
69 * @brief
                 Arduino setup function
70 */
71 void setup()
72 {
       // put your setup code here, to run once:
```

```
e.
38 **
   ** This may be due to may dynamic memory fragmentatio
   n.
40 ** Try defining "-DEI_CLASSIFIER_ALLOCATION_STATIC" in
   boards.local.txt (create
41 ** if it doesn't exist) and copy this file to
42 ** `<ARDUINO_CORE_INSTALL_PATH>/arduino/hardware/<mbed
   _core>/<core_version>/`.
43 **
44 ** See
45 ** (https://support.arduino.cc/hc/en-us/articles/36001
   2076960-Where-are-the-installed-cores-located-)
46 ** to find where Arduino installs cores on your machin
   e.
47 **
48 \,\, ** If the problem persists then there's not enough mem
   ory for this model and application.
49 */
50
51 /* Includes -----
   */
52 #include <FYP_Marvin_inferencing.h>
53
54
55
56 #ifdef CORE CM4
57 int pass;
58 int turn;
59 int infromM7 = D4;
60
61 #endif
62
63
64 int sound analog = A6;
65
66
67 /** Audio buffers, pointers and selectors */
68 typedef struct {
      signed short *buffers[2];
70
      unsigned char buf select;
71
      unsigned char buf_ready;
72
      unsigned int buf count;
       unsigned int n_samples;
73
74 } inference_t;
76 static inference_t inference;
77 static volatile bool record ready = false;
78 // static signed short *sampleBuffer;
79 static signed short sampleBuffer[2048];
80 static bool debug_nn = false; // Set this to true to se
   e e.g. features generated from the raw signal
81 static int print_results = -(EI_CLASSIFIER_SLICES_PER_M
   ODEL WINDOW);
82
83
85
86 /**
    * @brief
                 Arduino setup function
89 void setup()
90 {
91
     Serial.begin(115200);
92
    RPC.begin();
      // put your setup code here, to run once:
```

2/7

```
74
         Serial.begin(115200);
                                                                     94
 76
         Serial.println("Edge Impulse Inferencing Demo");
                                                                     95
 77
                                                                     96
         // summary of inferencing settings (from model_meta
 78
                                                                     97
    data.h)
 79
         ei_printf("Inferencing settings:\n");
                                                                     98
 80
         ei_printf("\tInterval: ");
                                                                     99
         ei_printf_float((float)EI_CLASSIFIER_INTERVAL_MS);
                                                                    100
 81
 82
         ei_printf(" ms.\n");
                                                                    101
         ei_printf("\tFrame size: %d\n", EI_CLASSIFIER_DSP_I
    NPUT FRAME SIZE);
                                                                         NPUT FRAME_SIZE);
         ei_printf("\tSample length: %d ms.\n", EI_CLASSIFIE
                                                                    103
    R RAW SAMPLE COUNT / 16);
        ei\_printf("\tNo. of classes: %d\n", sizeof(ei\_class)
                                                                    104
    ifier_inferencing_categories) /
                                                  sizeof(ei c
                                                                    105
    lassifier_inferencing_categories[0]));
 87
                                                                    106
 88
        run_classifier_init();
                                                                    107
        if (microphone_inference_start(EI_CLASSIFIER_SLICE_
    SIZE) == false) {
                                                                         SIZE) == false) {
 90
             ei_printf("ERR: Failed to setup audio sampling
                                                                    109
     \r\n");
                                                                         \r\n");
 91
             return;
                                                                                 return;
        }
                                                                    111
                                                                    112
                                                                    113
                                                                    114
                                                                               pass = 0;
                                                                    115
                                                                    116
                                                                    117
                                                                           #endif
 93 }
                                                                    118 }
 94
                                                                    119
 95 /**
                                                                    120 /**
     * @brief
                    Arduino main function. Runs the inferenc
                                                                         * @brief
    ing loop.
                                                                         ing loop.
 97 */
                                                                    122
                                                                          */
 98 void loop()
                                                                    123 void loop()
 99 {
                                                                    124
                                                                    125
                                                                    126
                                                                    127
                                                                    128
                                                                           if (turn == 0){
                                                                    129
                                                                    130
                                                                             pass = 0;
100
         bool m = microphone_inference_record();
                                                                    132
101
         if (!m) {
                                                                    133
                                                                             if (!m) {
102
             ei_printf("ERR: Failed to record audio...\n");
                                                                    134
103
             return:
                                                                    135
                                                                                 return:
104
                                                                    136
105
                                                                    137
106
         signal_t signal;
                                                                    138
107
         signal.total_length = EI_CLASSIFIER_SLICE_SIZE;
                                                                    139
         signal.get_data = &microphone_audio_signal_get_dat
108
                                                                    140
    a;
                                                                         a;
109
         ei_impulse_result_t result = {0};
                                                                    141
110
                                                                    142
         EI_IMPULSE_ERROR r = run_classifier_continuous(&sig
    nal, &result, debug_nn);
112
         if (r != EI_IMPULSE_OK) {
                                                                    144
             ei_printf("ERR: Failed to run classifier (%d)
113
                                                                    145
                                                                         \n", r);
     \n", r);
            return;
                                                                                 return;
```

```
RPC.println("Edge Impulse Inferencing Demo");
    // summary of inferencing settings (from model_meta
    ei_printf("Inferencing settings:\n");
    ei_printf("\tInterval: ");
    ei_printf_float((float)EI_CLASSIFIER_INTERVAL_MS);
    ei_printf(" ms.\n");
    ei_printf("\tFrame size: %d\n", EI_CLASSIFIER_DSP_I
    ei_printf("\tSample length: %d ms.\n", EI_CLASSIFIE
R RAW SAMPLE COUNT / 16);
    ei_printf("\tNo. of classes: %d\n", sizeof(ei_class
ifier_inferencing_categories) /
                                            sizeof(ei c
lassifier_inferencing_categories[0]));
    run_classifier_init();
    if (microphone_inference_start(EI_CLASSIFIER_SLICE_
        ei_printf("ERR: Failed to setup audio sampling
    #ifdef CORE_CM4
      pinMode(infromM7, INPUT);
      pinMode(sound_analog,INPUT);
               Arduino main function. Runs the inferenc
  turn = digitalRead(infromM7);
    auto res = RPC.call("setVar", pass).as<int>();
    bool m = microphone_inference_record();
       ei_printf("ERR: Failed to record audio...\n");
    signal_t signal;
    signal.total_length = EI_CLASSIFIER_SLICE_SIZE;
    signal.get_data = &microphone_audio_signal_get_dat
    ei_impulse_result_t result = {0};
    EI_IMPULSE_ERROR r = run_classifier_continuous(&sig
nal, &result, debug_nn);
    if (r != EI_IMPULSE_OK) {
       ei_printf("ERR: Failed to run classifier (%d)
```

[ix].label);

```
115
116
        if (++print_results >= (EI_CLASSIFIER_SLICES_PER_MO
117
    DEL_WINDOW)) {
118
            // print the predictions
119
            ei_printf("Predictions ");
120
            ei_printf("(DSP: %d ms., Classification: %d m
    s., Anomaly: %d ms.)",
121
                result.timing.dsp, result.timing.classifica
    tion, result.timing.anomaly);
            ei_printf(": \n");
            for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_CO</pre>
    UNT; ix++) {
                ei_printf(" %s: ", result.classification
124
```

## Computed Diff - Diff Checker

```
147
        if (++print_results >= (EI_CLASSIFIER_SLICES_PER_MO
    DEL_WINDOW)) {
150
151
152
            // print the predictions
153
            ei_printf("Predictions ");
            ei_printf("(DSP: %d ms., Classification: %d m
    s., Anomaly: %d ms.)",
               result.timing.dsp, result.timing.classifica
    tion, result.timing.anomaly);
            ei_printf(": \n");
158
159
            RPC.println("======");
160
            RPC.println("Sending From M4 core:");
161
            RPC.print("Predictions ");
163
            RPC.println("(DSP: " + String(result.timing.ds
    p) + " ms., Classification: " + String(result.timing.cl
    assification) + " ms., Anomaly: " + String(result.timin
    g.anomaly) + " ms.)");
165
           for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_CO</pre>
    UNT; ix++) {
               ei_printf(" %s: ", result.classification
167
    [ix].label);
```