EECE-4710 “IoT and TinyML”

Person Detection Example Application

Cristinel Ababei

MARQUETTE UNIVERSITY

BE THE DIFFERENCE.

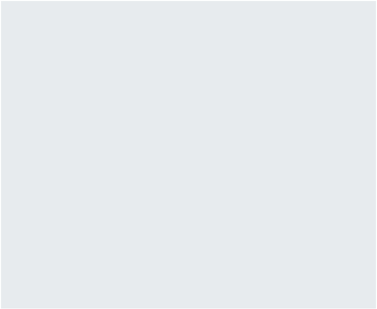
1

Person Detection

Application Architecture

2

✓



X

3

Person Detection using Transfer Learning Model

Code Walkthrough!

M:\arduino221\libraries\libraries\Arduino_TensorFlowLite\examples\person_detection\person_detection_v2.ino
(Available in the .zip file provided this week)

4

4

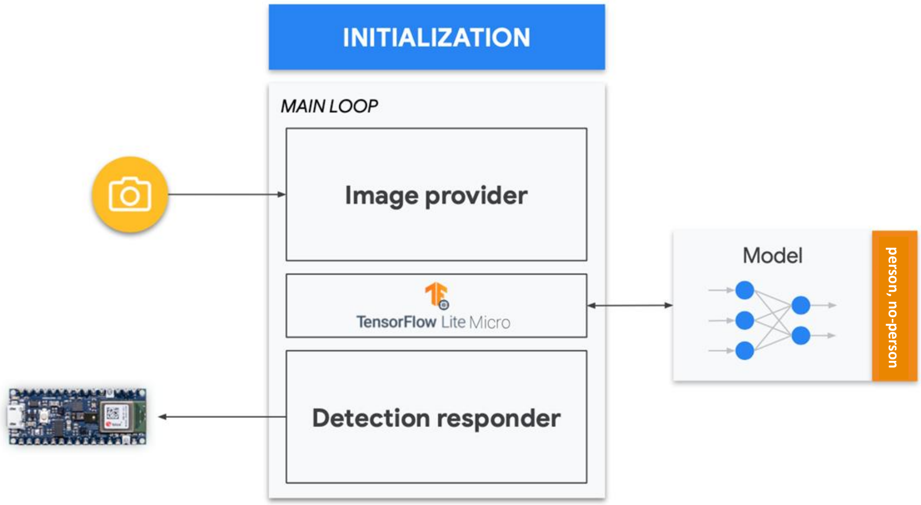
TensorFlow Lite Micro

TensorFlow Lite Micro - Paper

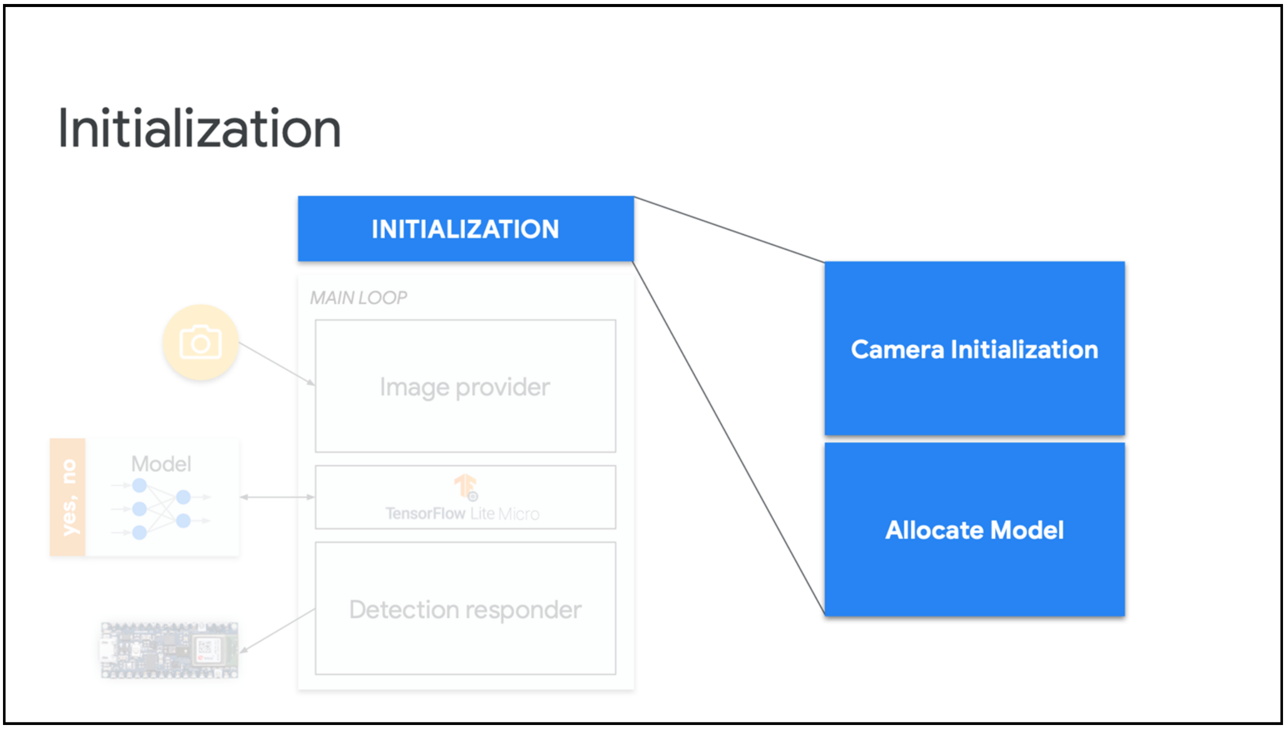
MLSys 2021: TensorFlow Lite Micro TFLM

5

Person Detection **Components**



6



Initialization

INITIALIZATION

MAIN LOOP

Image provider

Model

yes, no

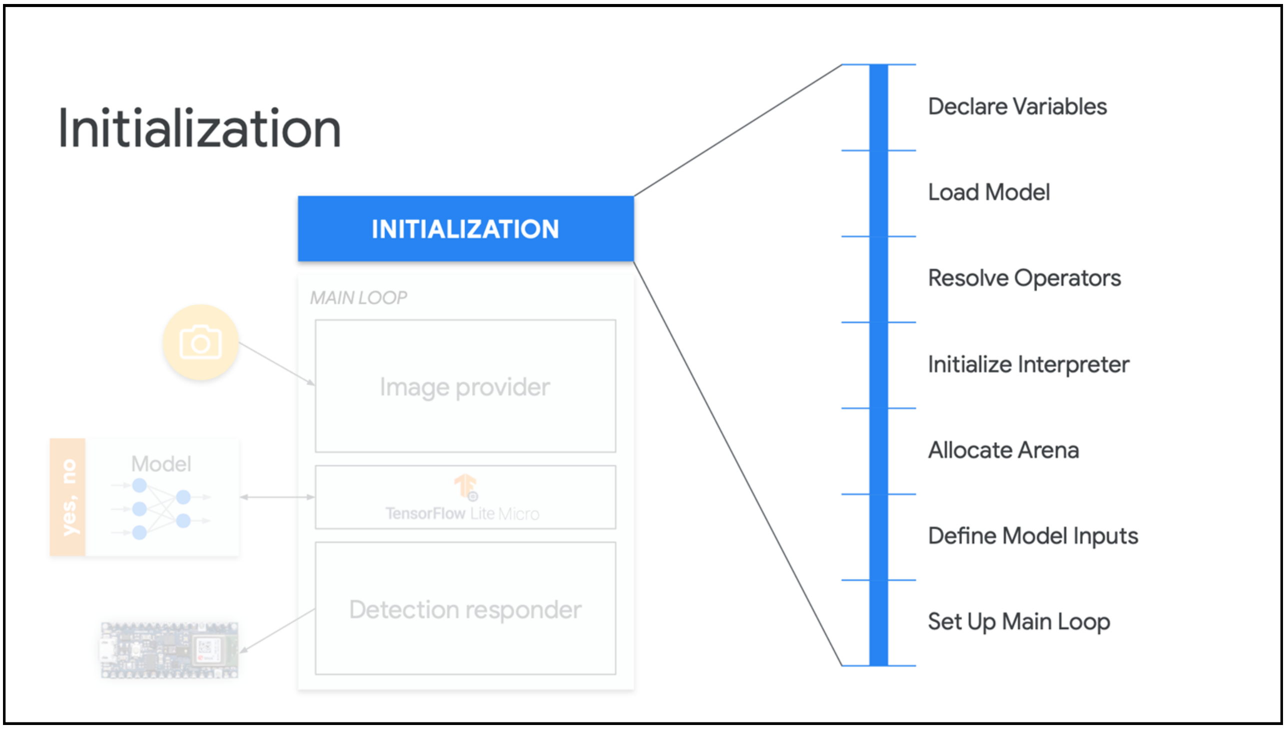
TensorFlow Lite Micro

Detection responder

Camera Initialization

Allocate Model

7



Initialization

INITIALIZATION

MAIN LOOP

Image provider

Model

yes, no

TensorFlow Lite Micro

Detection responder

Declare Variables

Load Model

Resolve Operators

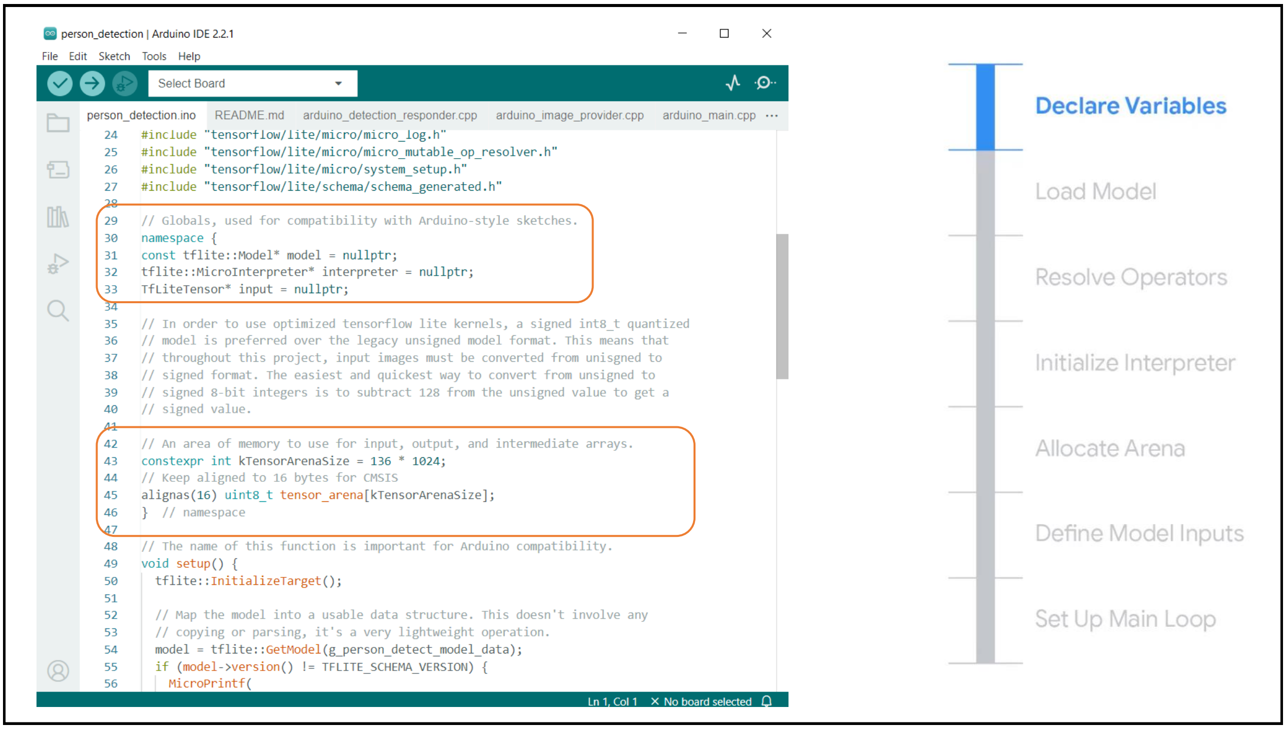
Initialize Interpreter

Allocate Arena

Define Model Inputs

Set Up Main Loop

8


person_detection | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Select Board
person_detection.ino README.md arduino_detection_responder.cpp arduino_image_provider.cpp arduino_main.cpp
Ln 1, Col 1 × No board selected

```
#include "tensorflow/lite/micro/micro_log.h"
#include "tensorflow/lite/micro/micro_mutable_op_resolver.h"
#include "tensorflow/lite/micro/system_setup.h"
#include "tensorflow/lite/schema/schema_generated.h"

// Globals, used for compatibility with Arduino-style sketches.
namespace {
const tflite::Model* model = nullptr;
tflite::MicroInterpreter* interpreter = nullptr;
TfLiteTensor* input = nullptr;

// In order to use optimized tensorflow lite kernels, a signed int8_t quantized
// model is preferred over the legacy unsigned model format. This means that
// throughout this project, input images must be converted from unisgned to
// signed format. The easiest and quickest way to convert from unsigned to
// signed 8-bit integers is to subtract 128 from the unsigned value to get a
// signed value.

// An area of memory to use for input, output, and intermediate arrays.
constexpr int kTensorArenaSize = 136 * 1024;
// Keep aligned to 16 bytes for CMSIS
alignas(16) uint8_t tensor_arena[kTensorArenaSize];
}  // namespace

// The name of this function is important for Arduino compatibility.
void setup() {
  tflite::InitializeTarget();

  // Map the model into a usable data structure. This doesn't involve any
  // copying or parsing, it's a very lightweight operation.
  model = tflite::GetModel(g_person_detect_model_data);
  if (model->version() != TFLITE_SCHEMA_VERSION) {
    MicroPrintf(
```

Declare Variables

Load Model

Resolve Operators

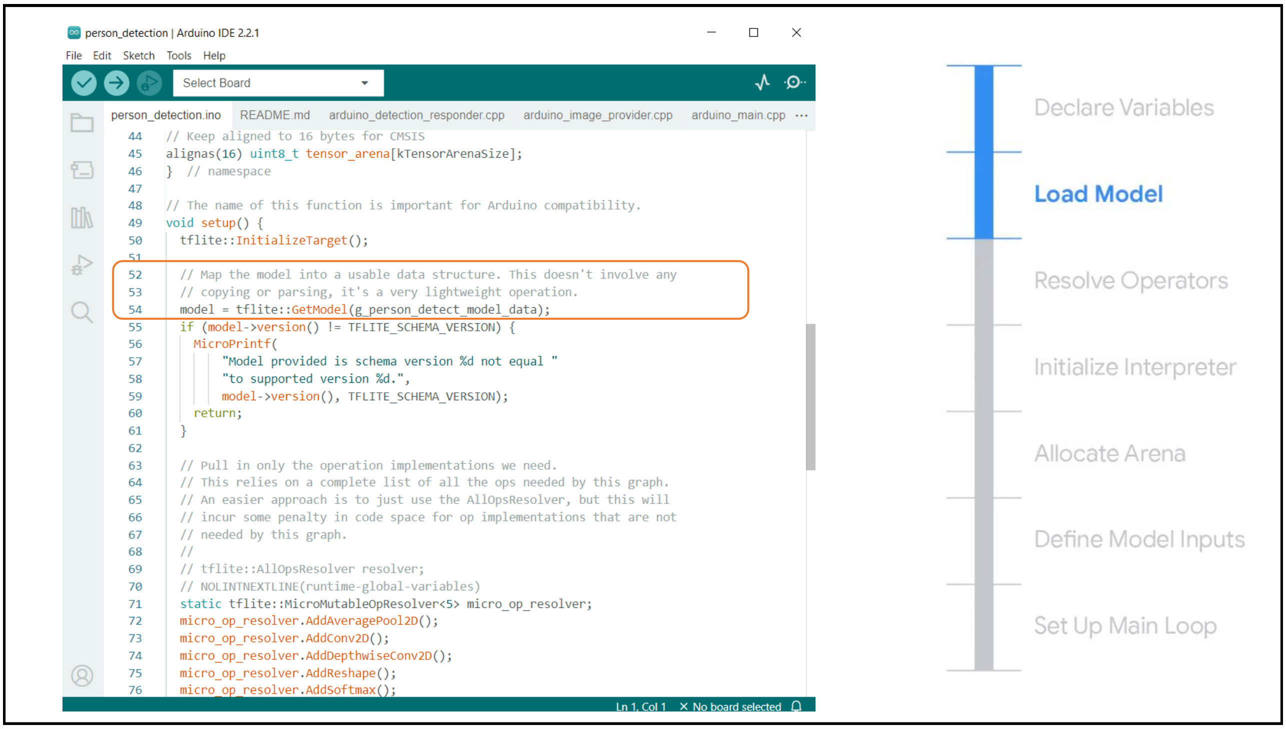
Initialize Interpreter

Allocate Arena

Define Model Inputs

Set Up Main Loop

9


person_detection | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Select Board
person_detection.ino README.md arduino_detection_responder.cpp arduino_image_provider.cpp arduino_main.cpp
Ln 1, Col 1 × No board selected

```
// Keep aligned to 16 bytes for CMSIS
alignas(16) uint8_t tensor_arena[kTensorArenaSize];
}  // namespace

// The name of this function is important for Arduino compatibility.
void setup() {
  tflite::InitializeTarget();

  // Map the model into a usable data structure. This doesn't involve any
  // copying or parsing, it's a very lightweight operation.
  model = tflite::GetModel(g_person_detect_model_data);
  if (model->version() != TFLITE_SCHEMA_VERSION) {
    MicroPrintf(
        "Model provided is schema version %d not equal "
        "to supported version %d.",
        model->version(), TFLITE_SCHEMA_VERSION);
    return;
  }

  // Pull in only the operation implementations we need.
  // This relies on a complete list of all the ops needed by this graph.
  // An easier approach is to just use the AllOpsResolver, but this will
  // incur some penalty in code space for op implementations that are not
  // needed by this graph.
  //
  // tflite::AllOpsResolver resolver;
  // NOLINTNEXTLINE(runtime-global-variables)
  static tflite::MicroMutableOpResolver<5> micro_op_resolver;
  micro_op_resolver.AddAveragePool2D();
  micro_op_resolver.AddConv2D();
  micro_op_resolver.AddDepthwiseConv2D();
  micro_op_resolver.AddReshape();
  micro_op_resolver.AddSoftmax();
```

Declare Variables

Load Model

Resolve Operators

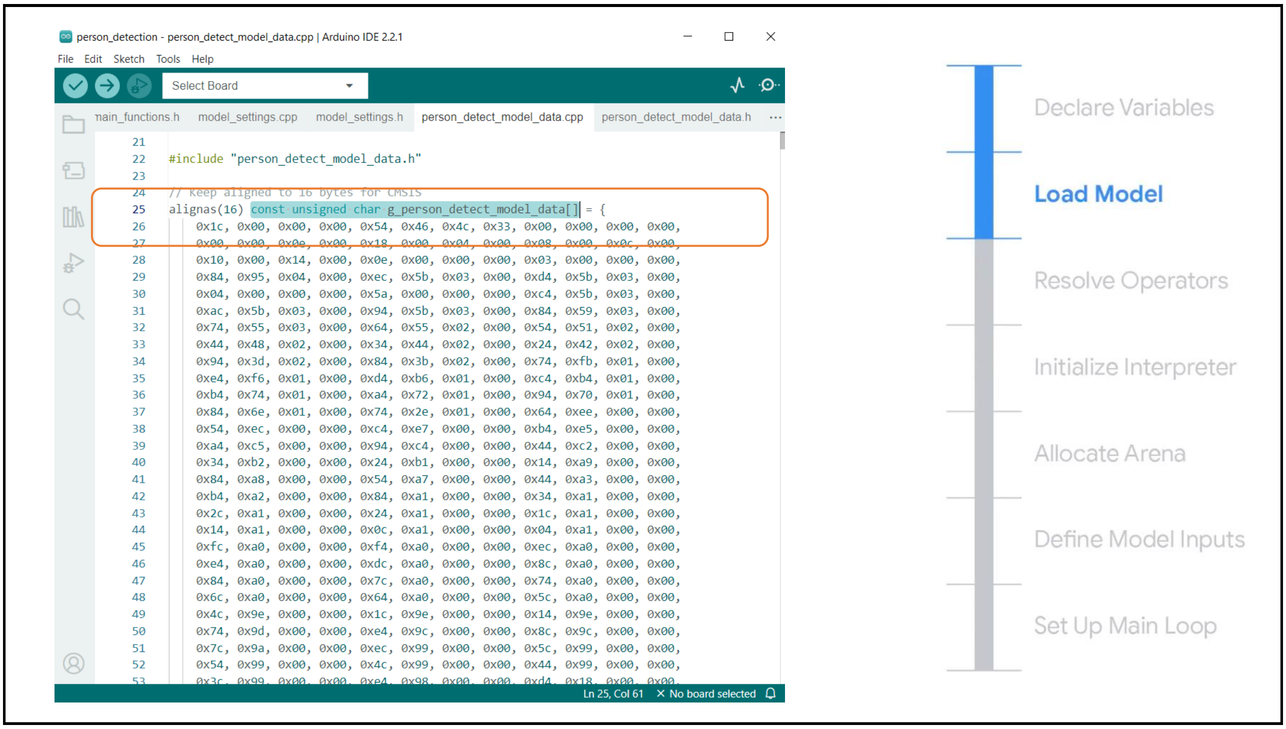
Initialize Interpreter

Allocate Arena

Define Model Inputs

Set Up Main Loop

10



person_detection - person_detect_model_data.cpp | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Select Board
nain_functions.h
model_settings.cpp
model_settings.h
person_detect_model_data.cpp
person_detect_model_data.h
Ln 25, Col 61 ✕ No board selected

```
#include "person_detect_model_data.h"

// Keep aligned to 16 bytes for CMSIS
alignas(16) const unsigned char g_person_detect_model_data[] = {
    0x1c, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x0e, 0x00, 0x18, 0x00, 0x04, 0x00, 0x08, 0x00, 0x0c, 0x00,
    0x10, 0x00, 0x14, 0x00, 0x0e, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00,
    0x84, 0x95, 0x04, 0x00, 0xec, 0x5b, 0x03, 0x00, 0xd4, 0x5b, 0x03, 0x00,
    0x04, 0x00, 0x00, 0x00, 0x5a, 0x00, 0x00, 0x00, 0xc4, 0x5b, 0x03, 0x00,
    0xac, 0x5b, 0x03, 0x00, 0x94, 0x5b, 0x03, 0x00, 0x84, 0x59, 0x03, 0x00,
    0x74, 0x55, 0x03, 0x00, 0x64, 0x55, 0x02, 0x00, 0x54, 0x51, 0x02, 0x00,
    0x44, 0x48, 0x02, 0x00, 0x34, 0x44, 0x02, 0x00, 0x24, 0x42, 0x02, 0x00,
    0x94, 0x3d, 0x02, 0x00, 0x84, 0x3b, 0x02, 0x00, 0x74, 0xfb, 0x01, 0x00,
    0xe4, 0xf6, 0x01, 0x00, 0xd4, 0xb6, 0x01, 0x00, 0xc4, 0xb4, 0x01, 0x00,
    0xb4, 0x74, 0x01, 0x00, 0xa4, 0x72, 0x01, 0x00, 0x94, 0x70, 0x01, 0x00,
    0x84, 0x6e, 0x01, 0x00, 0x74, 0x2e, 0x01, 0x00, 0x64, 0xee, 0x00, 0x00,
    0x54, 0xec, 0x00, 0x00, 0xc4, 0xe7, 0x00, 0x00, 0xb4, 0xe5, 0x00, 0x00,
    0xa4, 0xc5, 0x00, 0x00, 0x94, 0xc4, 0x00, 0x00, 0x44, 0xc2, 0x00, 0x00,
    0x34, 0xb2, 0x00, 0x00, 0x24, 0xb1, 0x00, 0x00, 0x14, 0xa9, 0x00, 0x00,
    0x84, 0xa8, 0x00, 0x00, 0x54, 0xa7, 0x00, 0x00, 0x44, 0xa3, 0x00, 0x00,
    0xb4, 0xa2, 0x00, 0x00, 0x84, 0xa1, 0x00, 0x00, 0x34, 0xa1, 0x00, 0x00,
    0x2c, 0xa1, 0x00, 0x00, 0x24, 0xa1, 0x00, 0x00, 0x1c, 0xa1, 0x00, 0x00,
    0x14, 0xa1, 0x00, 0x00, 0x0c, 0xa1, 0x00, 0x00, 0x04, 0xa1, 0x00, 0x00,
    0xfc, 0xa0, 0x00, 0x00, 0xf4, 0xa0, 0x00, 0x00, 0xec, 0xa0, 0x00, 0x00,
    0xe4, 0xa0, 0x00, 0x00, 0xdc, 0xa0, 0x00, 0x00, 0x8c, 0xa0, 0x00, 0x00,
    0x84, 0xa0, 0x00, 0x00, 0x7c, 0xa0, 0x00, 0x00, 0x74, 0xa0, 0x00, 0x00,
    0x6c, 0xa0, 0x00, 0x00, 0x64, 0xa0, 0x00, 0x00, 0x5c, 0xa0, 0x00, 0x00,
    0x4c, 0x9e, 0x00, 0x00, 0x1c, 0x9e, 0x00, 0x00, 0x14, 0x9e, 0x00, 0x00,
    0x74, 0x9d, 0x00, 0x00, 0xe4, 0x9c, 0x00, 0x00, 0x8c, 0x9c, 0x00, 0x00,
    0x7c, 0x9a, 0x00, 0x00, 0xec, 0x99, 0x00, 0x00, 0x5c, 0x99, 0x00, 0x00,
    0x54, 0x99, 0x00, 0x00, 0x4c, 0x99, 0x00, 0x00, 0x44, 0x99, 0x00, 0x00,
    0x3c, 0x99, 0x00, 0x00, 0xe4, 0x98, 0x00, 0x00, 0xd4, 0x18, 0x00, 0x00,
```

Declare Variables

Load Model

Resolve Operators

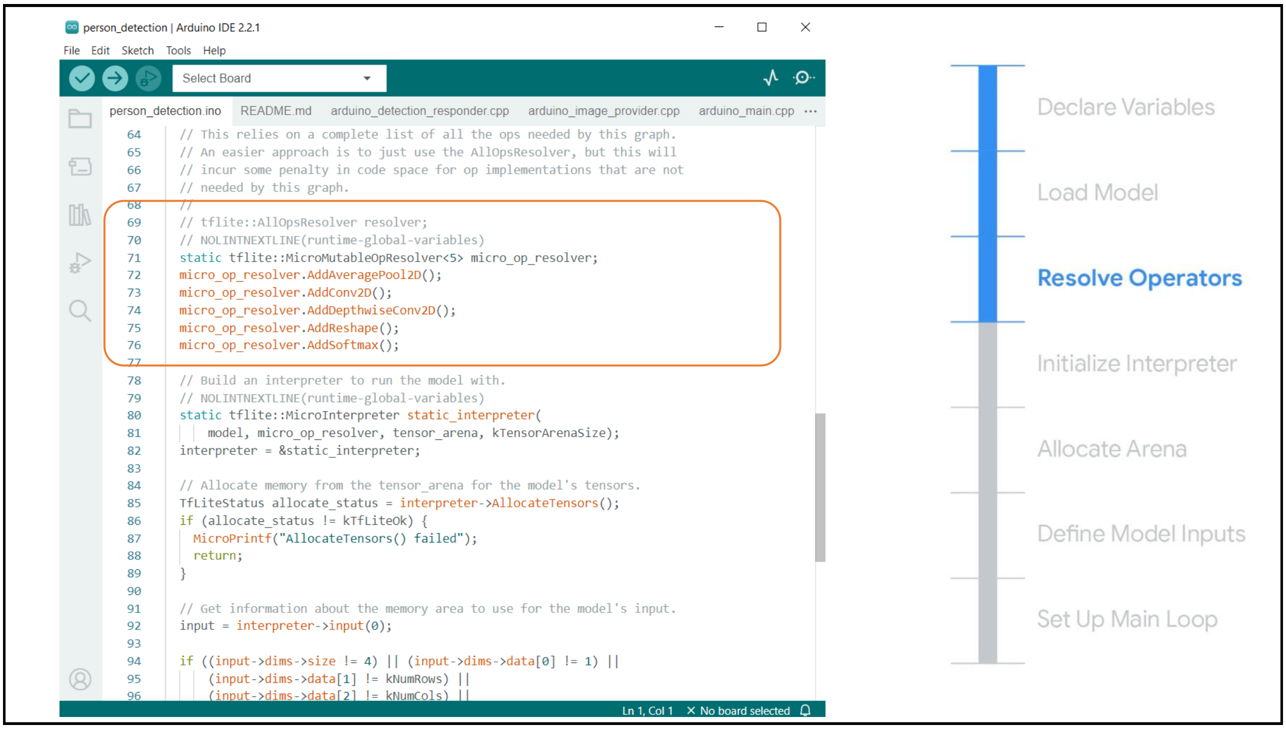
Initialize Interpreter

Allocate Arena

Define Model Inputs

Set Up Main Loop

11



person_detection | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Select Board
person_detection.ino
README.md
arduino_detection_responder.cpp
arduino_image_provider.cpp
arduino_main.cpp
Ln 1, Col 1 ✕ No board selected

```
// This relies on a complete list of all the ops needed by this graph.
// An easier approach is to just use the AllOpsResolver, but this will
// incur some penalty in code space for op implementations that are not
// needed by this graph.
//
// tflite::AllOpsResolver resolver;
// NOLINTNEXTLINE(runtime-global-variables)
static tflite::MicroMutableOpResolver<5> micro_op_resolver;
micro_op_resolver.AddAveragePool2D();
micro_op_resolver.AddConv2D();
micro_op_resolver.AddDepthwiseConv2D();
micro_op_resolver.AddReshape();
micro_op_resolver.AddSoftmax();

// Build an interpreter to run the model with.
// NOLINTNEXTLINE(runtime-global-variables)
static tflite::MicroInterpreter static_interpreter(
    model, micro_op_resolver, tensor_arena, kTensorArenaSize);
interpreter = &static_interpreter;

// Allocate memory from the tensor_arena for the model's tensors.
TfLiteStatus allocate_status = interpreter->AllocateTensors();
if (allocate_status != kTfLiteOk) {
  MicroPrintf("AllocateTensors() failed");
  return;
}

// Get information about the memory area to use for the model's input.
input = interpreter->input(0);

if ((input->dims->size != 4) || (input->dims->data[0] != 1) ||
    (input->dims->data[1] != kNumRows) ||
    (input->dims->data[2] != kNumCols) ||
```

Declare Variables

Load Model

Resolve Operators

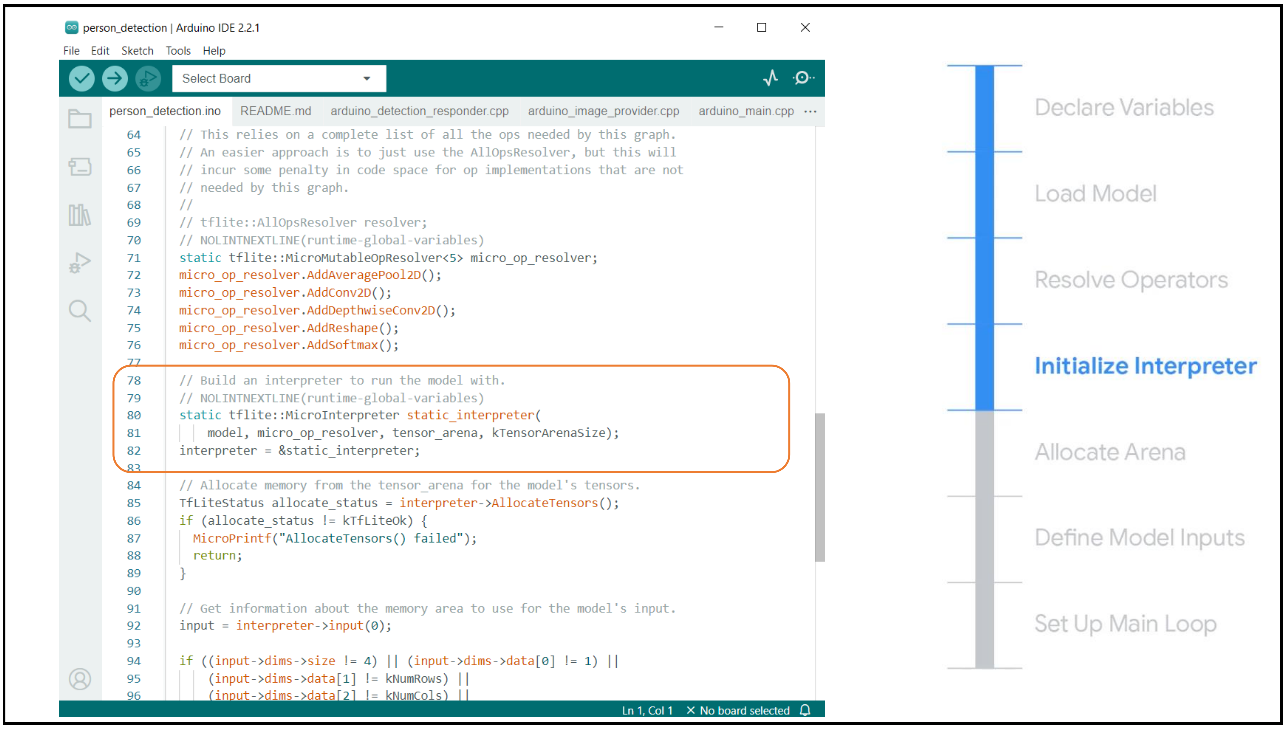
Initialize Interpreter

Allocate Arena

Define Model Inputs

Set Up Main Loop

12


person_detection | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Select Board
person_detection.ino
README.md
arduino_detection_responder.cpp
arduino_image_provider.cpp
arduino_main.cpp
Ln 1, Col 1
No board selected

```
  // This relies on a complete list of all the ops needed by this graph.
  // An easier approach is to just use the AllOpsResolver, but this will
  // incur some penalty in code space for op implementations that are not
  // needed by this graph.
  //
  // tflite::AllOpsResolver resolver;
  // NOLINTNEXTLINE(runtime-global-variables)
  static tflite::MicroMutableOpResolver<5> micro_op_resolver;
  micro_op_resolver.AddAveragePool2D();
  micro_op_resolver.AddConv2D();
  micro_op_resolver.AddDepthwiseConv2D();
  micro_op_resolver.AddReshape();
  micro_op_resolver.AddSoftmax();

  // Build an interpreter to run the model with.
  // NOLINTNEXTLINE(runtime-global-variables)
  static tflite::MicroInterpreter static_interpreter(
      model, micro_op_resolver, tensor_arena, kTensorArenaSize);
  interpreter = &static_interpreter;

  // Allocate memory from the tensor_arena for the model's tensors.
  TfLiteStatus allocate_status = interpreter->AllocateTensors();
  if (allocate_status != kTfLiteOk) {
    MicroPrintf("AllocateTensors() failed");
    return;
  }

  // Get information about the memory area to use for the model's input.
  input = interpreter->input(0);

  if ((input->dims->size != 4) || (input->dims->data[0] != 1) ||
      (input->dims->data[1] != kNumRows) ||
      (input->dims->data[2] != kNumCols) ||
```

Declare Variables

Load Model

Resolve Operators

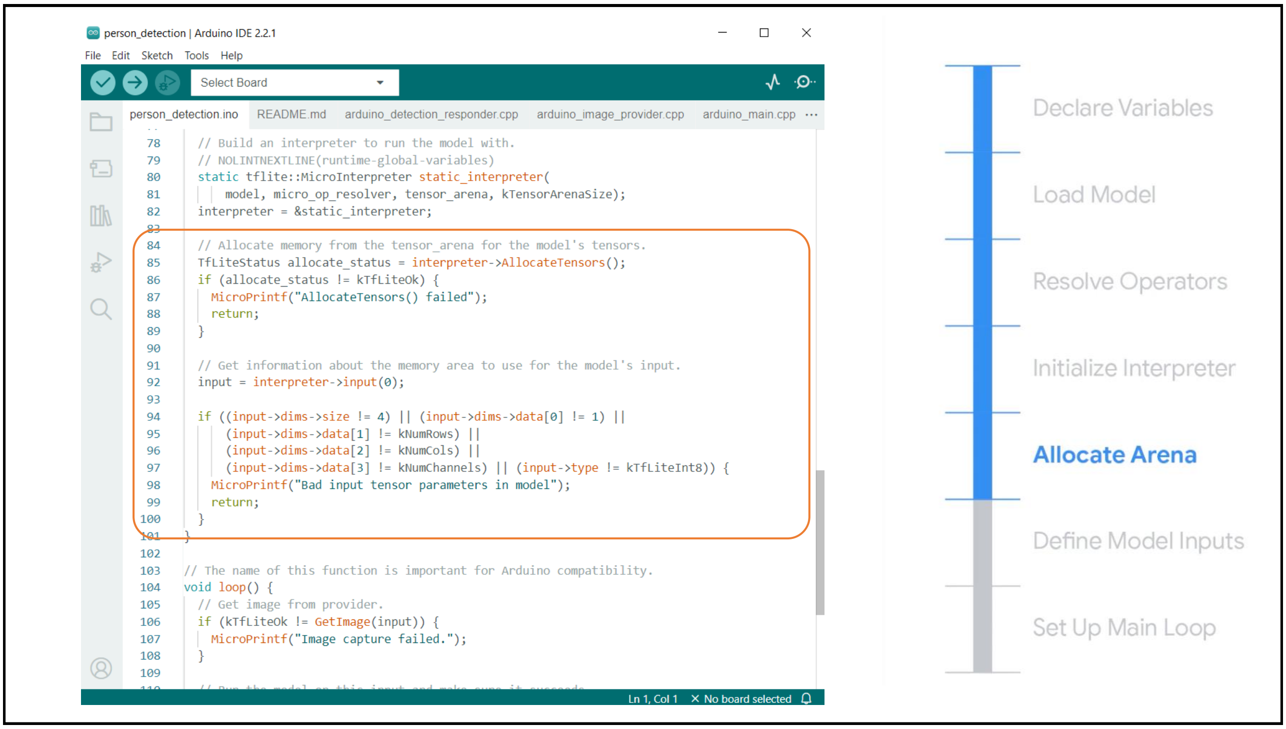
Initialize Interpreter

Allocate Arena

Define Model Inputs

Set Up Main Loop

13


person_detection | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Select Board
person_detection.ino
README.md
arduino_detection_responder.cpp
arduino_image_provider.cpp
arduino_main.cpp
Ln 1, Col 1
No board selected

```
  // Build an interpreter to run the model with.
  // NOLINTNEXTLINE(runtime-global-variables)
  static tflite::MicroInterpreter static_interpreter(
      model, micro_op_resolver, tensor_arena, kTensorArenaSize);
  interpreter = &static_interpreter;

  // Allocate memory from the tensor_arena for the model's tensors.
  TfLiteStatus allocate_status = interpreter->AllocateTensors();
  if (allocate_status != kTfLiteOk) {
    MicroPrintf("AllocateTensors() failed");
    return;
  }

  // Get information about the memory area to use for the model's input.
  input = interpreter->input(0);

  if ((input->dims->size != 4) || (input->dims->data[0] != 1) ||
      (input->dims->data[1] != kNumRows) ||
      (input->dims->data[2] != kNumCols) ||
      (input->dims->data[3] != kNumChannels) || (input->type != kTfLiteInt8)) {
    MicroPrintf("Bad input tensor parameters in model");
    return;
  }
}

// The name of this function is important for Arduino compatibility.
void loop() {
  // Get image from provider.
  if (kTfLiteOk != GetImage(input)) {
    MicroPrintf("Image capture failed.");
  }

  // Run the model on this input and make sure it succeeds.
```

Declare Variables

Load Model

Resolve Operators

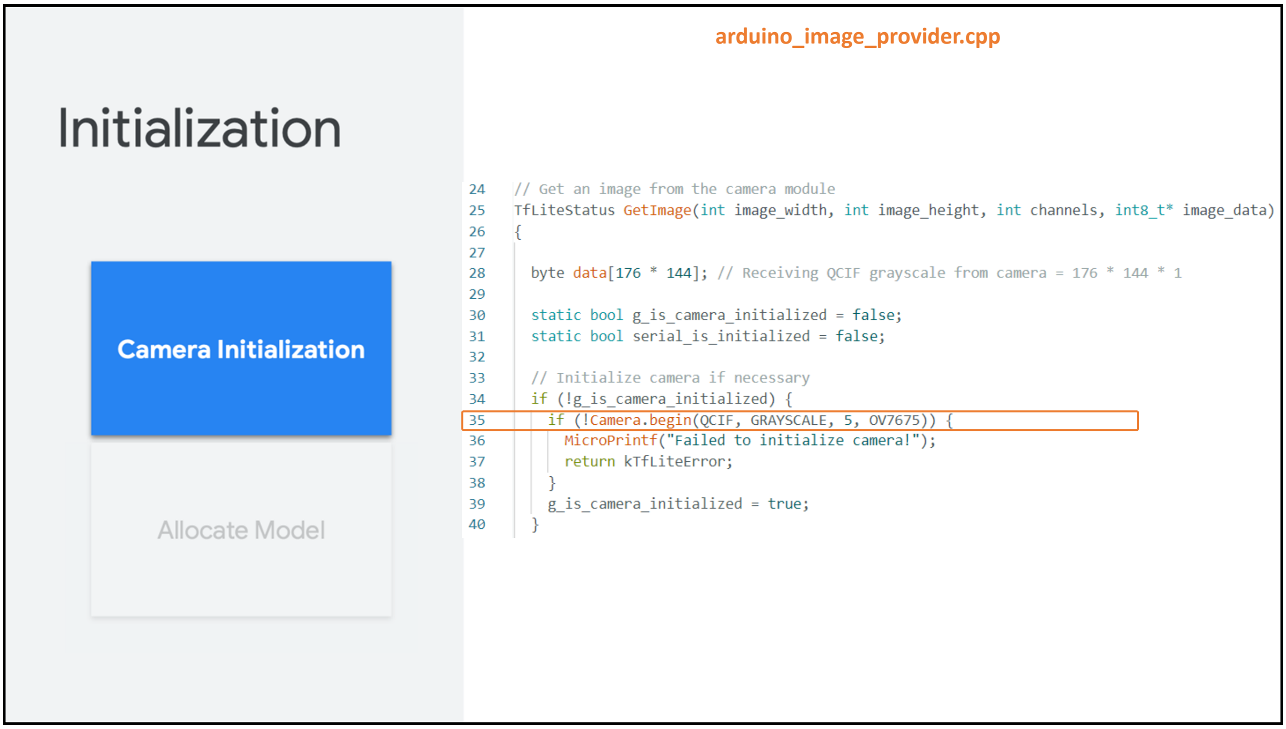
Initialize Interpreter

Allocate Arena

Define Model Inputs

Set Up Main Loop

14



Initialization

Camera Initialization

Allocate Model

arduino_image_provider.cpp

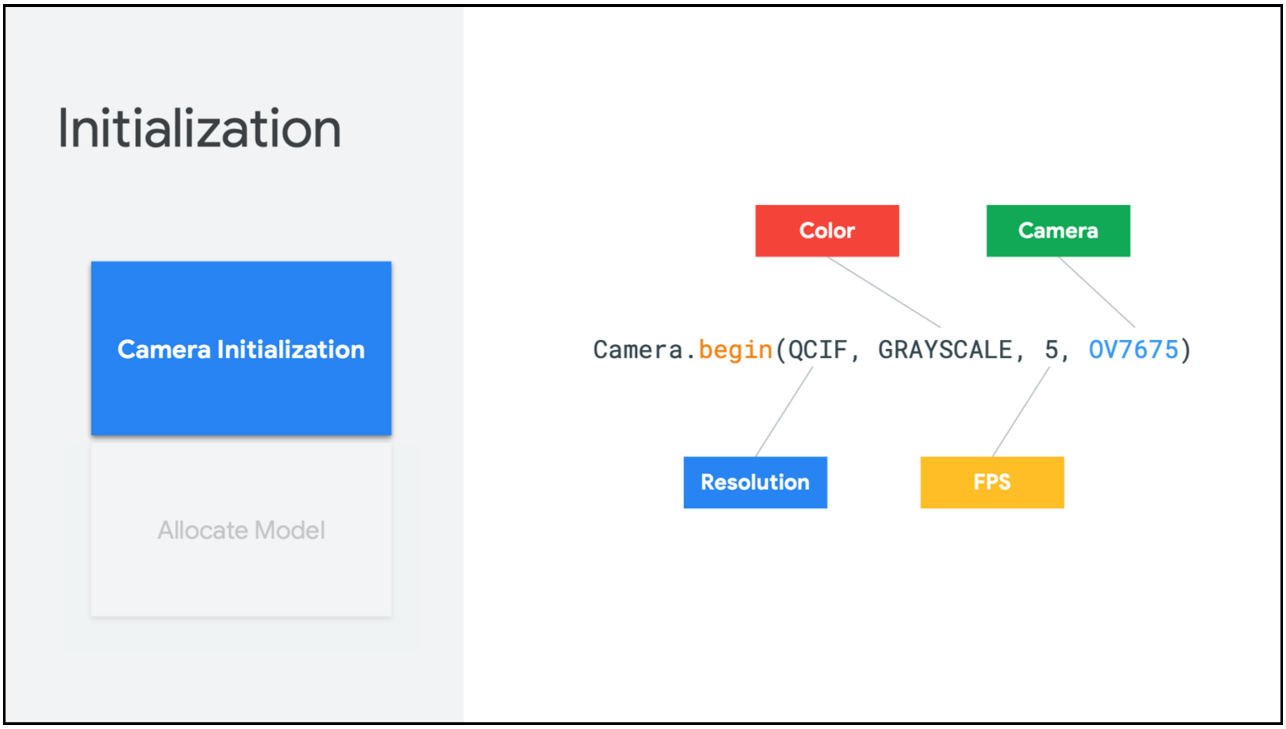
```
// Get an image from the camera module
TfLiteStatus GetImage(int image_width, int image_height, int channels, int8_t* image_data)
{

  byte data[176 * 144]; // Receiving QCIF grayscale from camera = 176 * 144 * 1

  static bool g_is_camera_initialized = false;
  static bool serial_is_initialized = false;

  // Initialize camera if necessary
  if (!g_is_camera_initialized) {
    if (!Camera.begin(QCIF, GRAYSCALE, 5, OV7675)) {
      MicroPrintf("Failed to initialize camera!");
      return kTfLiteError;
    }
    g_is_camera_initialized = true;
  }
```

15



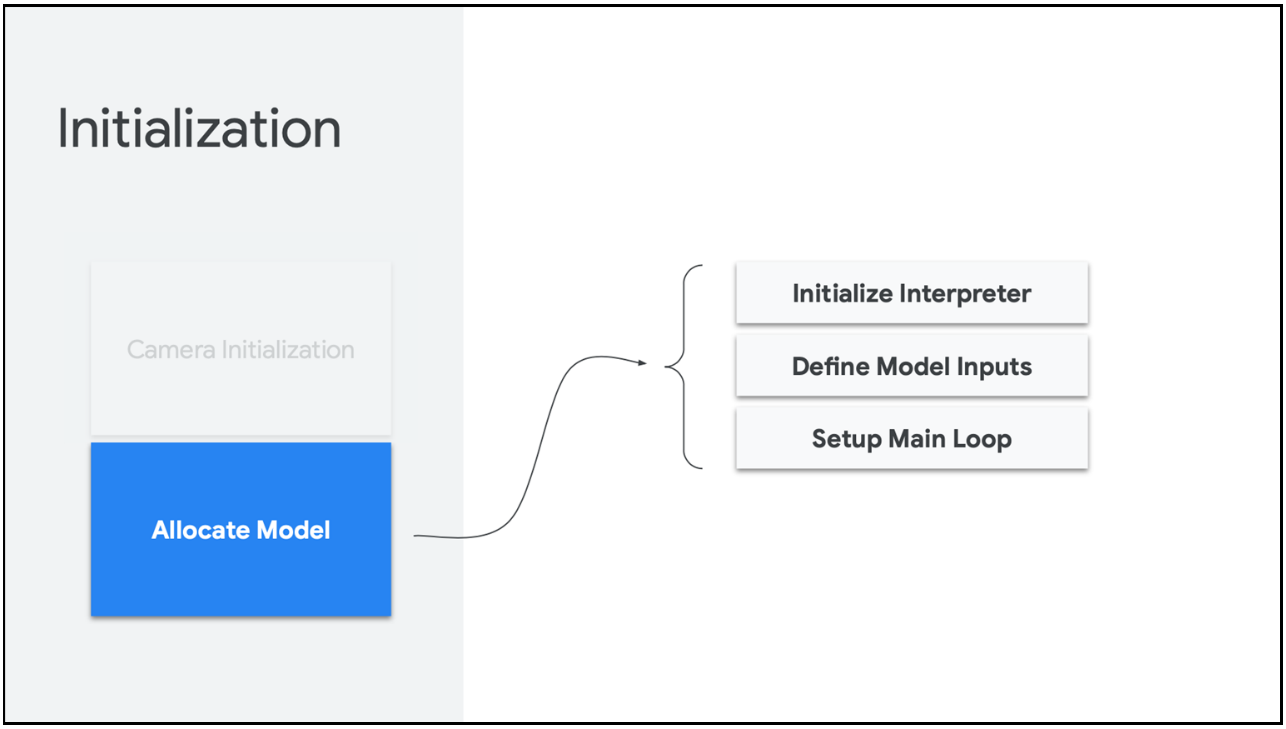
Initialization

Camera Initialization

Allocate Model

```
Camera.begin(QCIF, GRAYSCALE, 5, OV7675)
```

16



Initialization

Camera Initialization

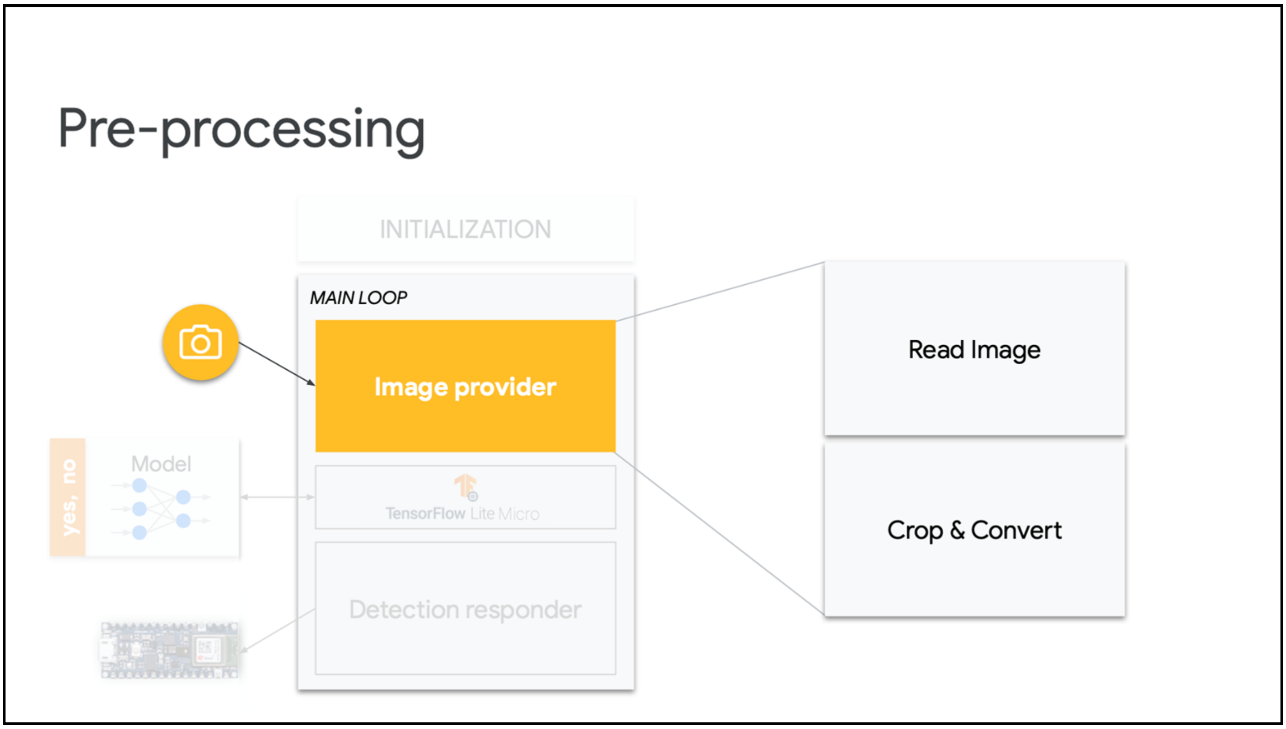
Allocate Model

Initialize Interpreter

Define Model Inputs

Setup Main Loop

17



Pre-processing

INITIALIZATION

MAIN LOOP

Image provider

TensorFlow Lite Micro

Detection responder

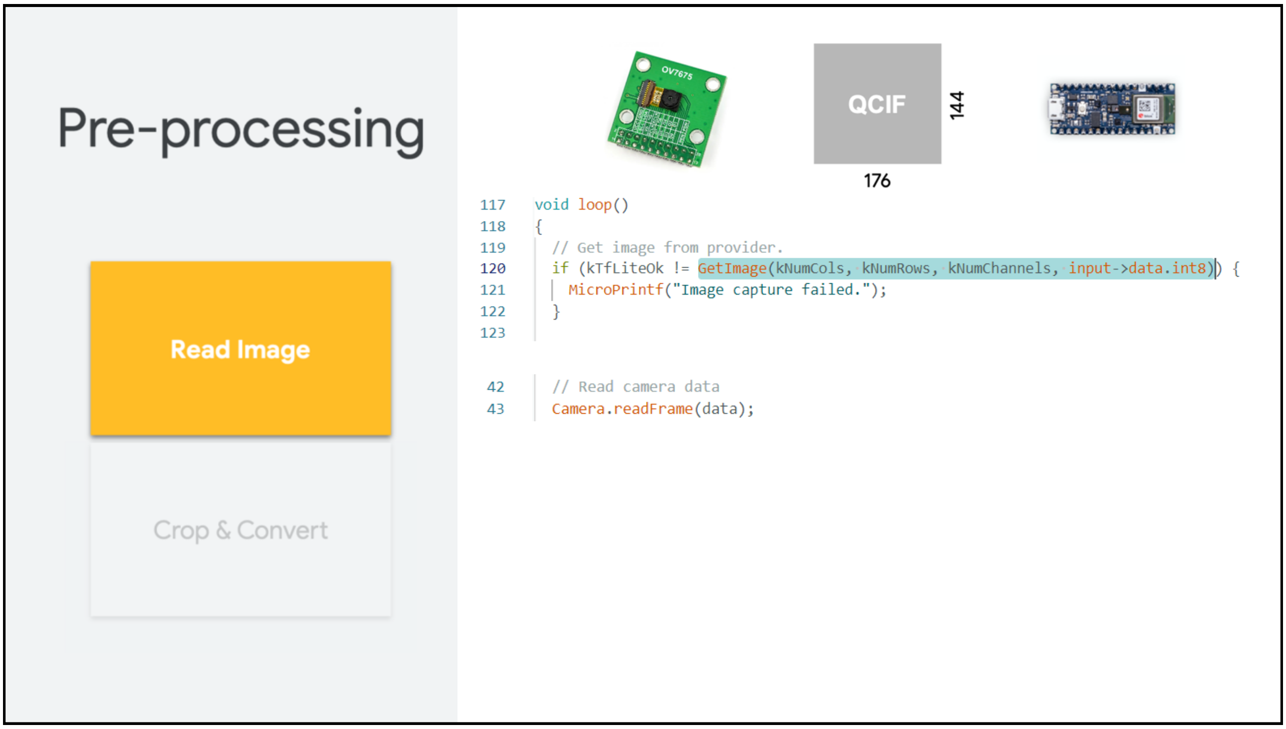
yes, no

Model

Read Image

Crop & Convert

18



Pre-processing

Read Image

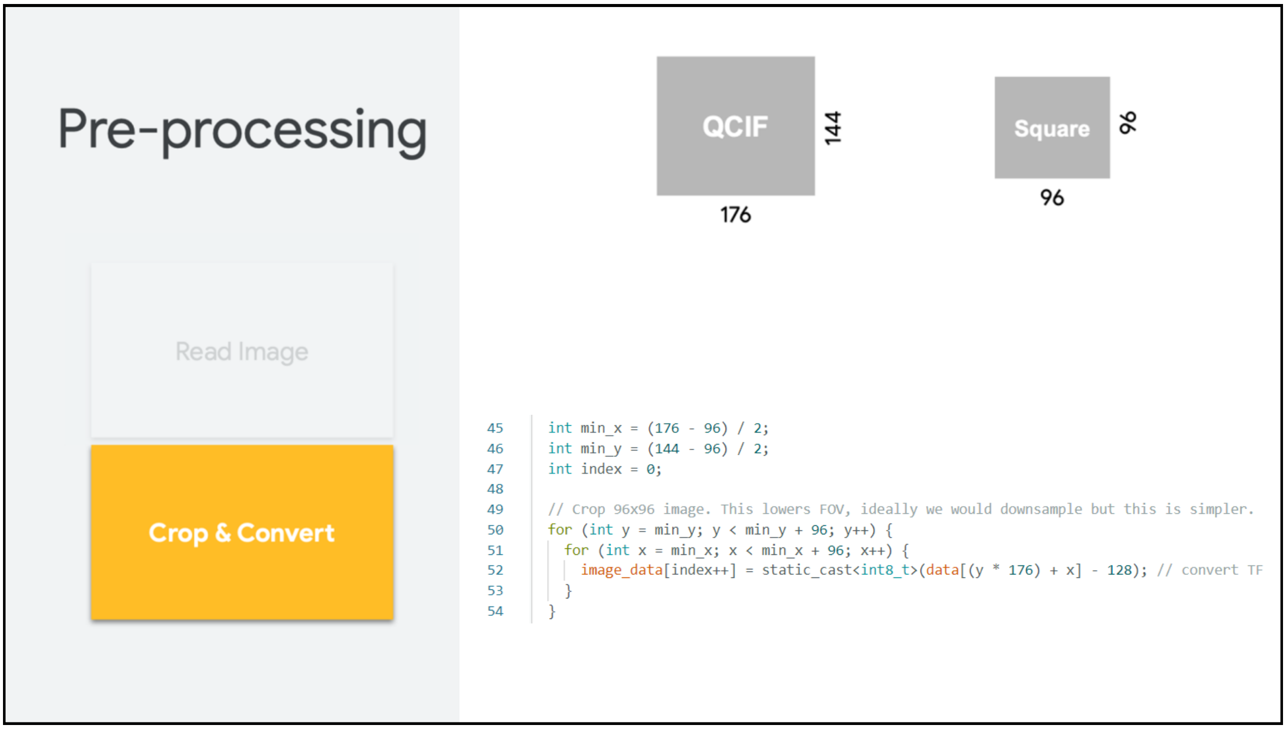
Crop & Convert

```
void loop()
{
  // Get image from provider.
  if (kTfLiteOk != GetImage(kNumCols, kNumRows, kNumChannels, input->data.int8)) {
    MicroPrintf("Image capture failed.");
  }

```

```
  // Read camera data
  Camera.readFrame(data);
```

19



Pre-processing

Read Image

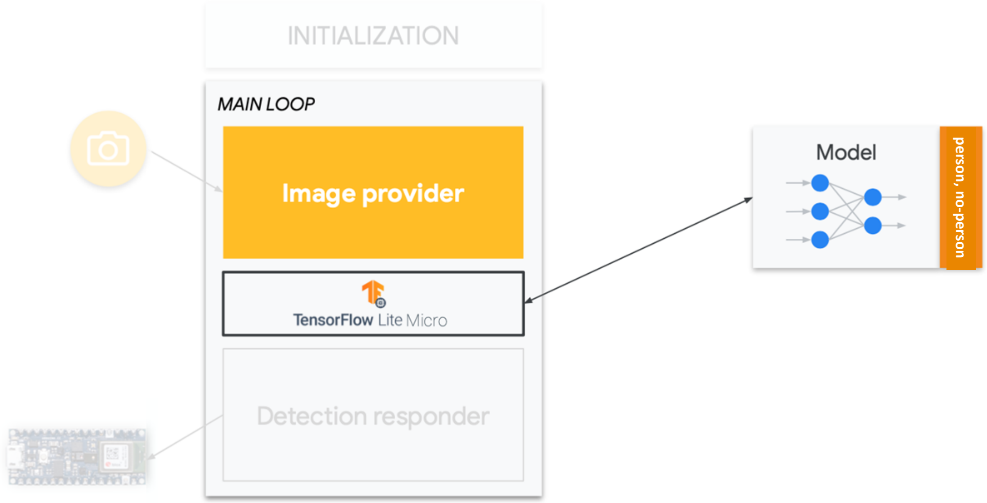
Crop & Convert

```
int min_x = (176 - 96) / 2;
int min_y = (144 - 96) / 2;
int index = 0;

// Crop 96x96 image. This lowers FOV, ideally we would downsample but this is simpler.
for (int y = min_y; y < min_y + 96; y++) {
  for (int x = min_x; x < min_x + 96; x++) {
    image_data[index++] = static_cast<int8_t>(data[(y * 176) + x] - 128); // convert TF
  }
}
```

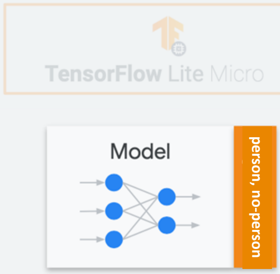
20

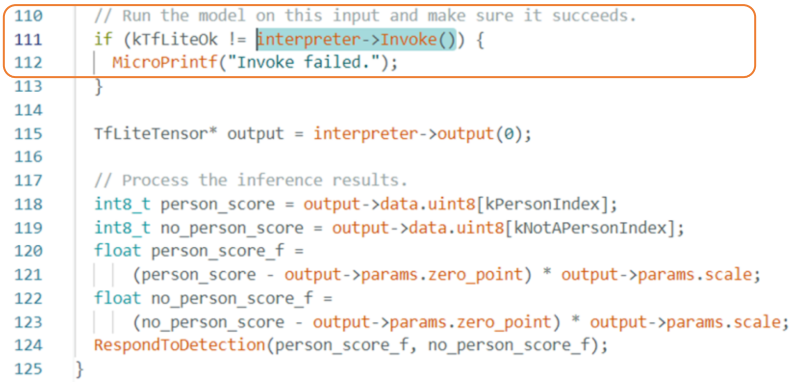
Interpreter + Model



21

Interpreter + Model



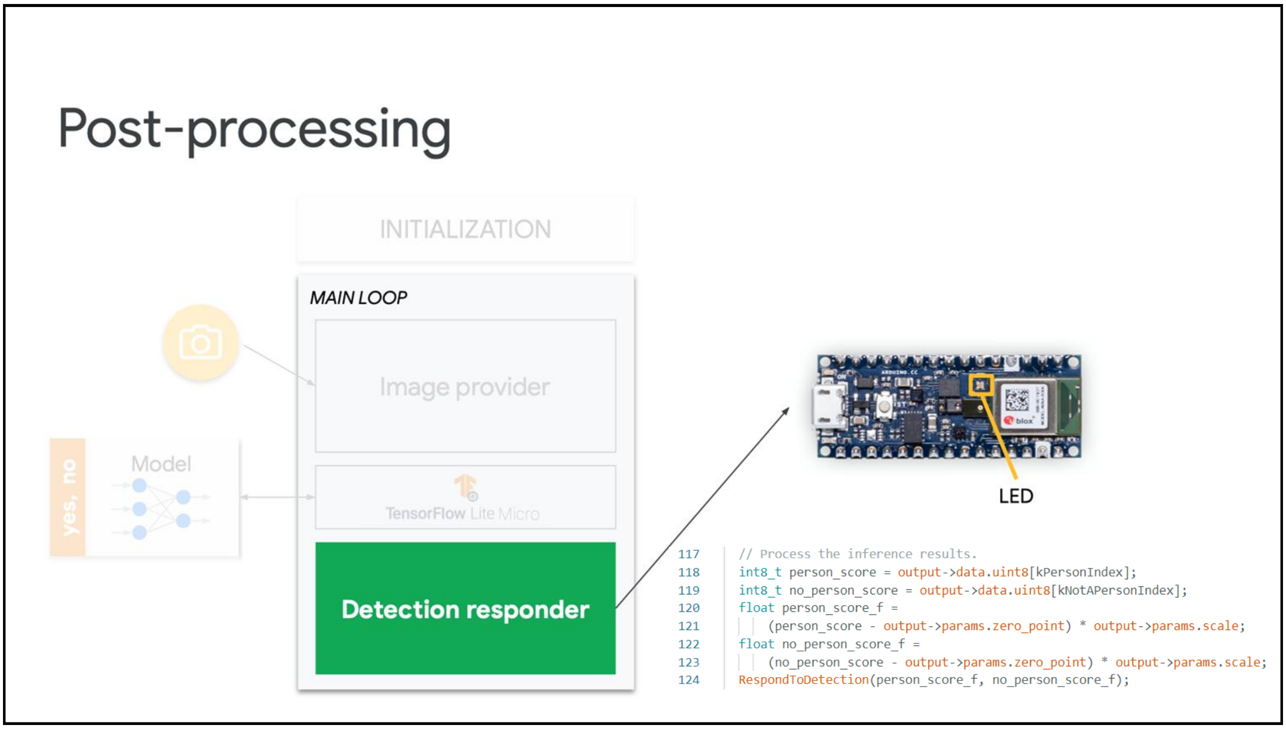


```
    // Run the model on this input and make sure it succeeds.
    if (kTfLiteOk != interpreter->Invoke()) {
      MicroPrintf("Invoke failed.");
    }

    TfLiteTensor* output = interpreter->output(0);

    // Process the inference results.
    int8_t person_score = output->data.uint8[kPersonIndex];
    int8_t no_person_score = output->data.uint8[kNotAPersonIndex];
    float person_score_f =
        (person_score - output->params.zero_point) * output->params.scale;
    float no_person_score_f =
        (no_person_score - output->params.zero_point) * output->params.scale;
    RespondToDetection(person_score_f, no_person_score_f);
  }
```

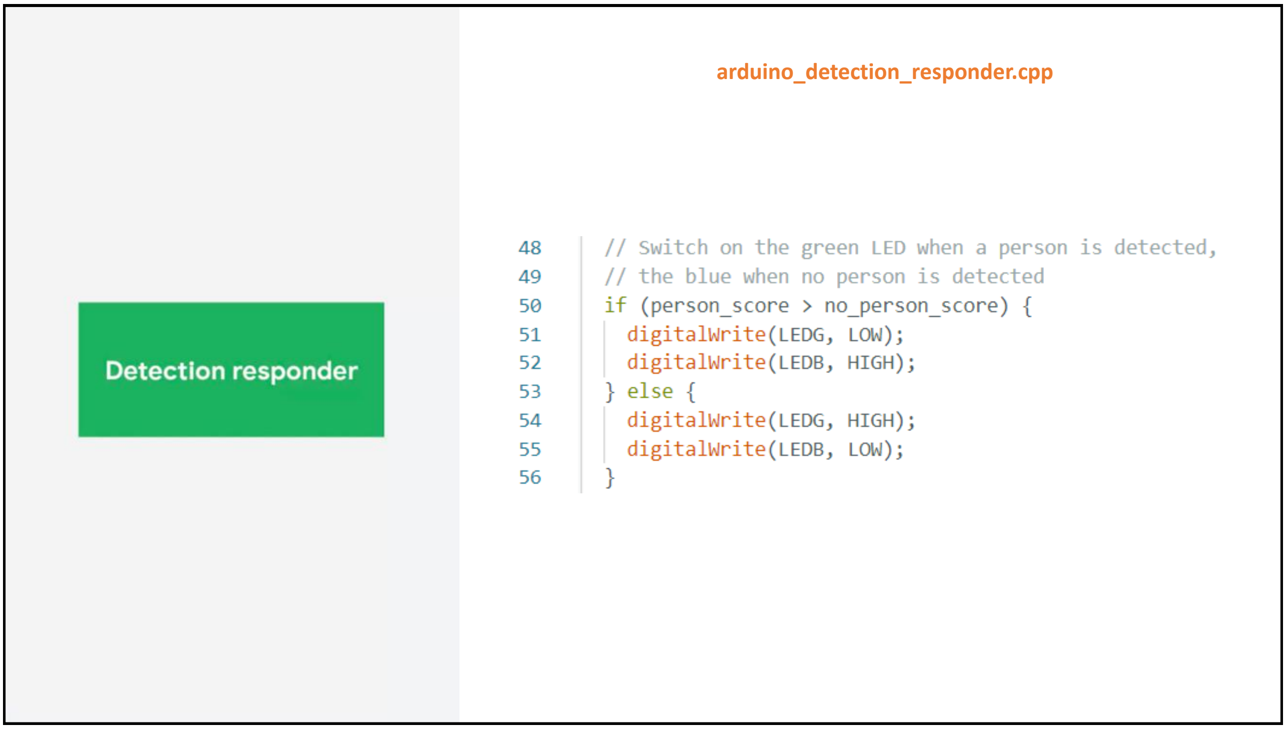
22



Post-processing

```
// Process the inference results.
int8_t person_score = output->data.uint8[kPersonIndex];
int8_t no_person_score = output->data.uint8[kNotAPersonIndex];
float person_score_f =
    (person_score - output->params.zero_point) * output->params.scale;
float no_person_score_f =
    (no_person_score - output->params.zero_point) * output->params.scale;
RespondToDetection(person_score_f, no_person_score_f);
```

23



Detection responder

arduino_detection_responder.cpp

```
// Switch on the green LED when a person is detected,
// the blue when no person is detected
if (person_score > no_person_score) {
  digitalWrite(LEDG, LOW);
  digitalWrite(LEDB, HIGH);
} else {
  digitalWrite(LEDG, HIGH);
  digitalWrite(LEDB, LOW);
}
```

24

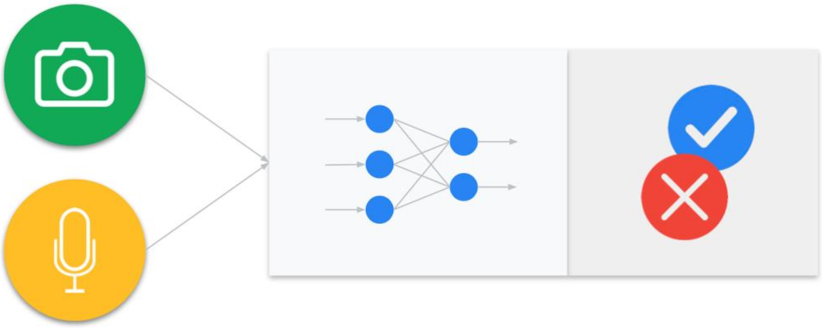
Person Detection (Optional)

Multi-Tenancy

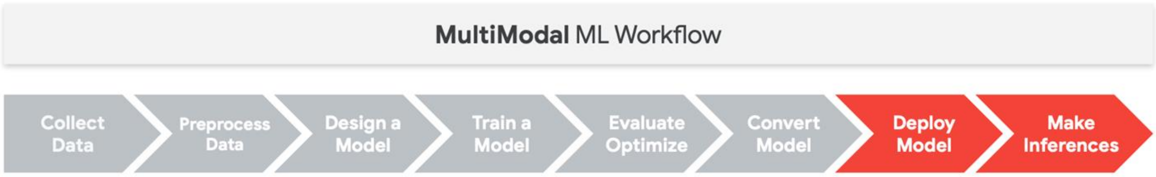
25

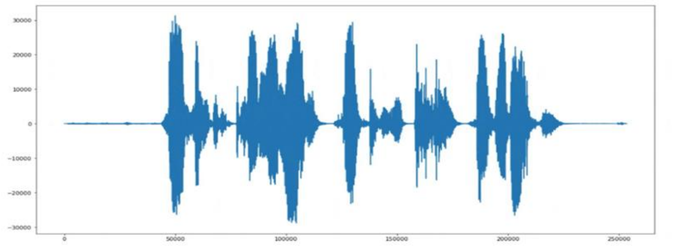
25

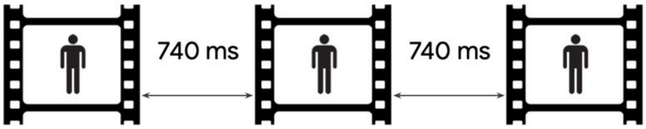
MultiModal



26





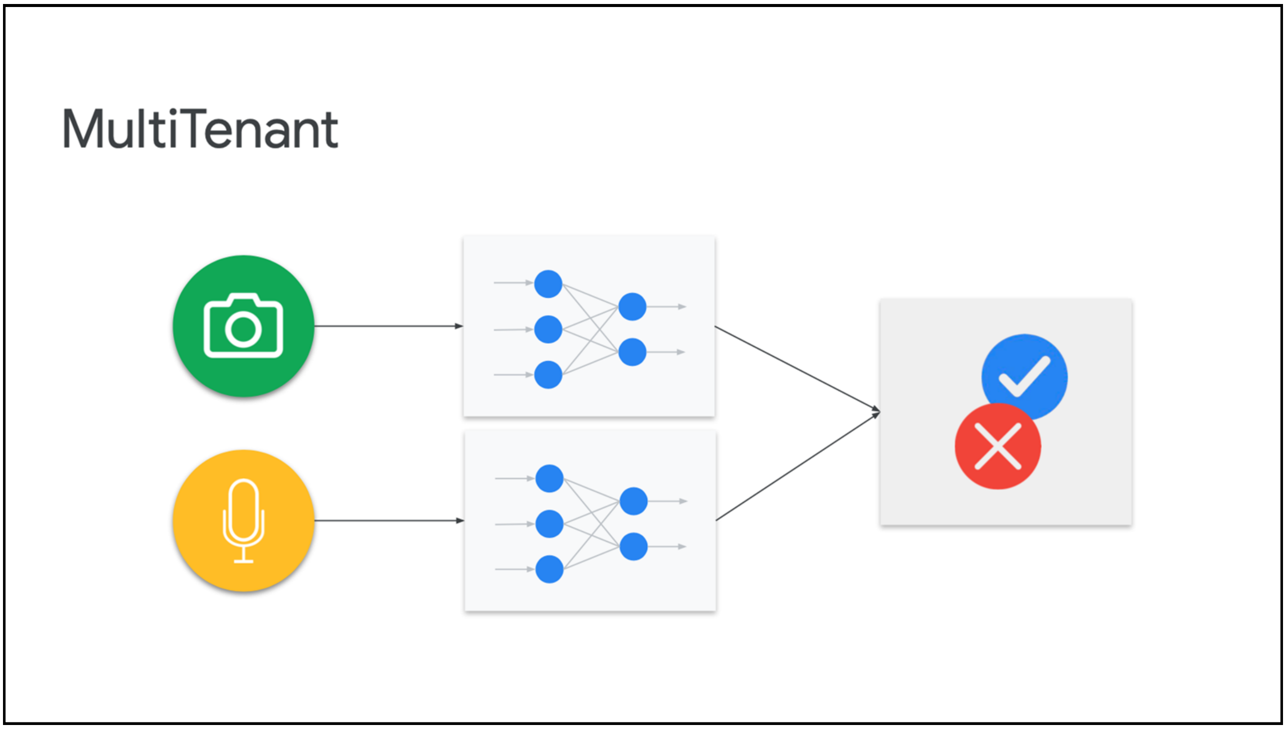


27

Example Person Detection Application

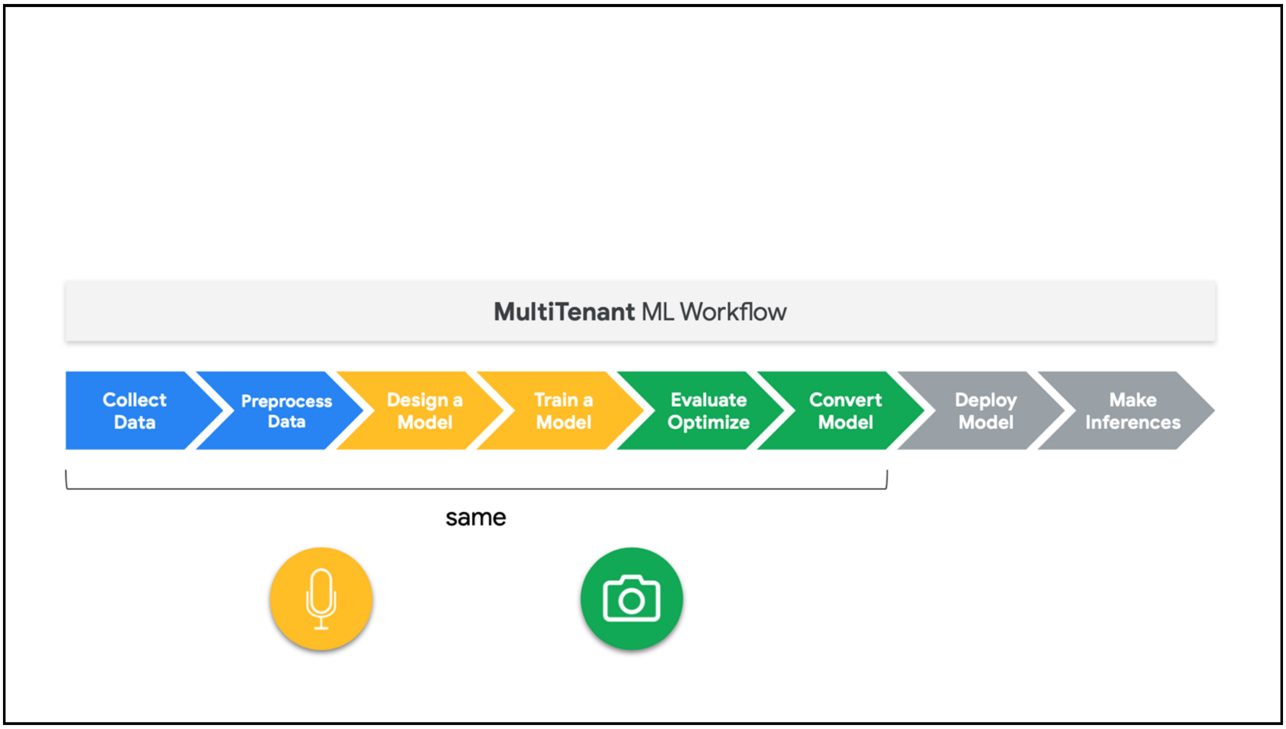
- Contact-free elevator control that enforces mask wearing
- Requires both **keyword spotting** and **mask detection**

28



MultiTenant

29



MultiTenant ML Workflow

Collect Data

Preprocess Data

Design a Model

Train a Model

Evaluate Optimize

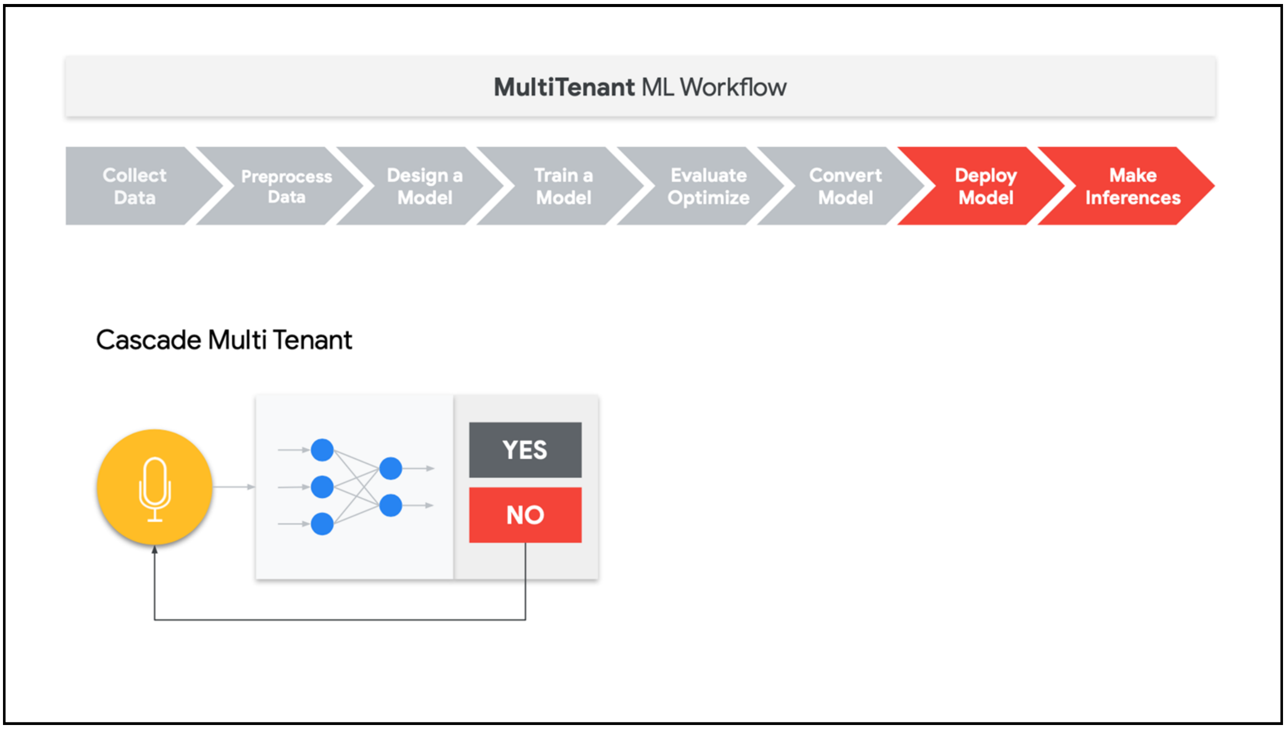
Convert Model

Deploy Model

Make Inferences

same

30

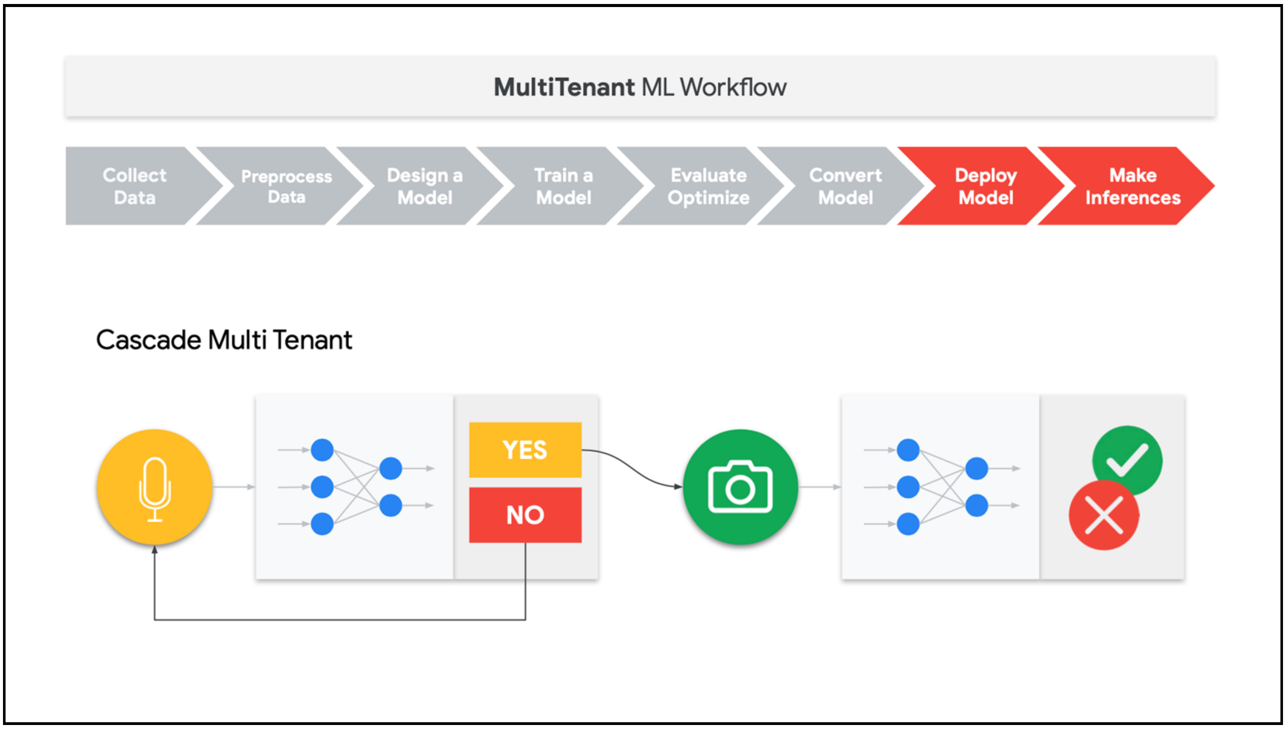


MultiTenant ML Workflow

Collect Data → Preprocess Data → Design a Model → Train a Model → Evaluate Optimize → Convert Model → Deploy Model → Make Inferences

Cascade Multi Tenant

31

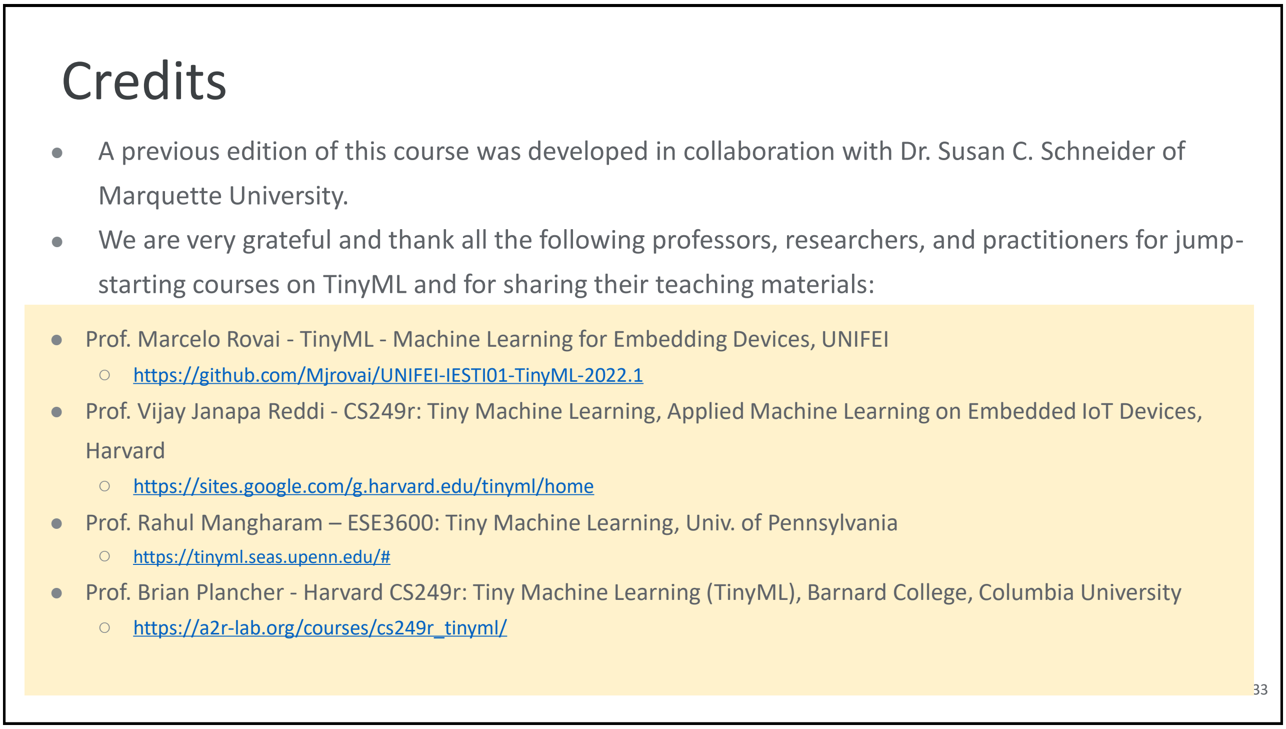


MultiTenant ML Workflow

Collect Data → Preprocess Data → Design a Model → Train a Model → Evaluate Optimize → Convert Model → Deploy Model → Make Inferences

Cascade Multi Tenant

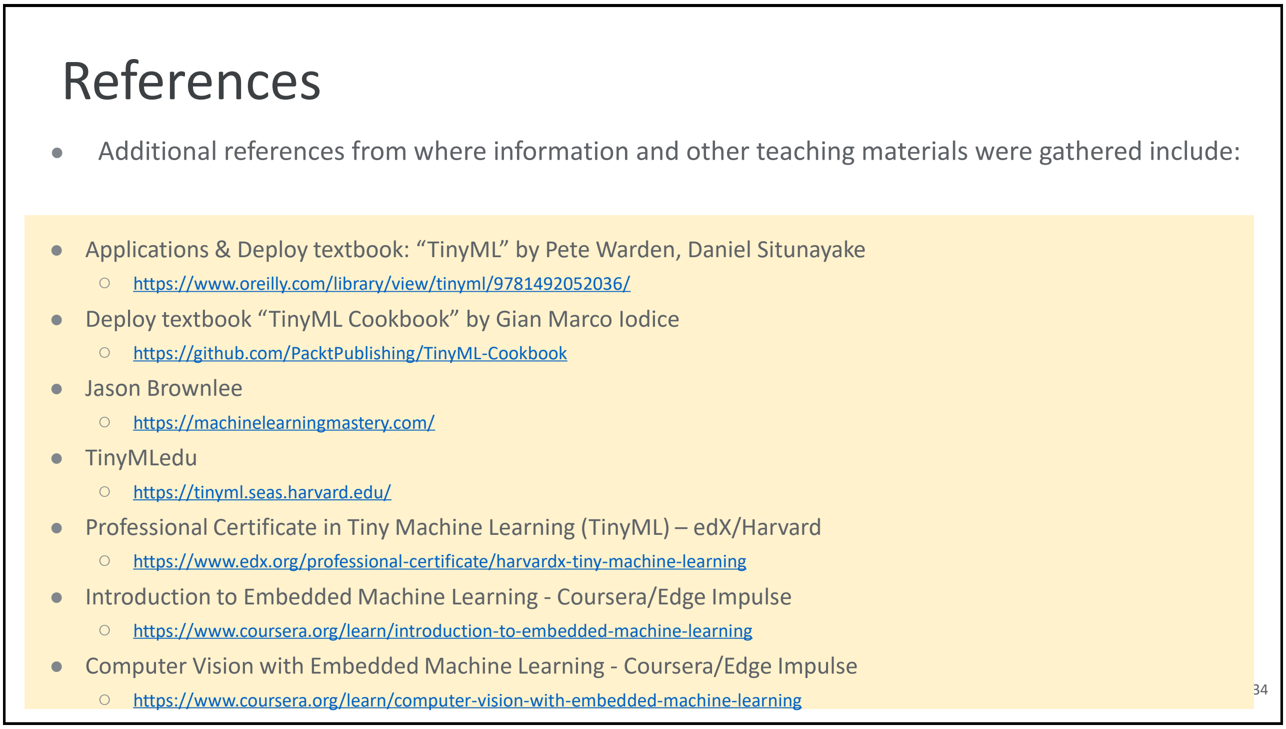
32



Credits

- A previous edition of this course was developed in collaboration with Dr. Susan C. Schneider of Marquette University.
- We are very grateful and thank all the following professors, researchers, and practitioners for jump-starting courses on TinyML and for sharing their teaching materials:

- Prof. Marcelo Rovai - TinyML - Machine Learning for Embedding Devices, UNIFEI
  - https://github.com/Mjrovai/UNIFEI-IESTI01-TinyML-2022.1
- Prof. Vijay Janapa Reddi - CS249r: Tiny Machine Learning, Applied Machine Learning on Embedded IoT Devices, Harvard
  - https://sites.google.com/g.harvard.edu/tinyml/home
- Prof. Rahul Mangharam – ESE3600: Tiny Machine Learning, Univ. of Pennsylvania
  - https://tinyml.seas.upenn.edu/#
- Prof. Brian Plancher - Harvard CS249r: Tiny Machine Learning (TinyML), Barnard College, Columbia University
  - https://a2r-lab.org/courses/cs249r_tinyml/



References

- Additional references from where information and other teaching materials were gathered include:

- Applications & Deploy textbook: “TinyML” by Pete Warden, Daniel Situnayake
  - https://www.oreilly.com/library/view/tinyml/9781492052036/
- Deploy textbook “TinyML Cookbook” by Gian Marco Iodice
  - https://github.com/PacktPublishing/TinyML-Cookbook
- Jason Brownlee
  - https://machinelearningmastery.com/
- TinyMLedu
  - https://tinyml.seas.harvard.edu/
- Professional Certificate in Tiny Machine Learning (TinyML) – edX/Harvard
  - https://www.edx.org/professional-certificate/harvardx-tiny-machine-learning
- Introduction to Embedded Machine Learning - Coursera/Edge Impulse
  - https://www.coursera.org/learn/introduction-to-embedded-machine-learning
- Computer Vision with Embedded Machine Learning - Coursera/Edge Impulse
  - https://www.coursera.org/learn/computer-vision-with-embedded-machine-learning