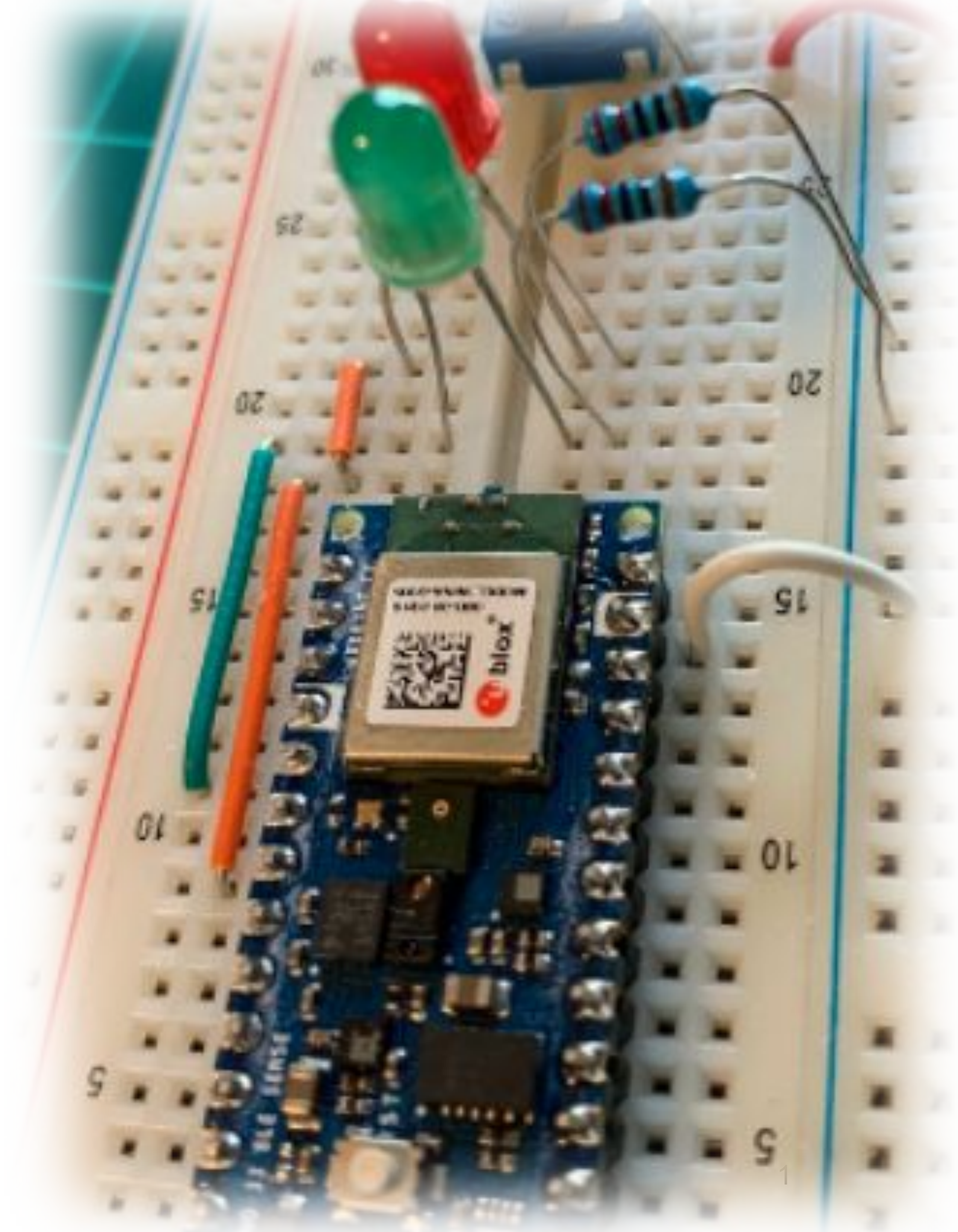
IESTI01 – TinyML

Embedded Machine Learning

25a. Person Detection (VWW) Application

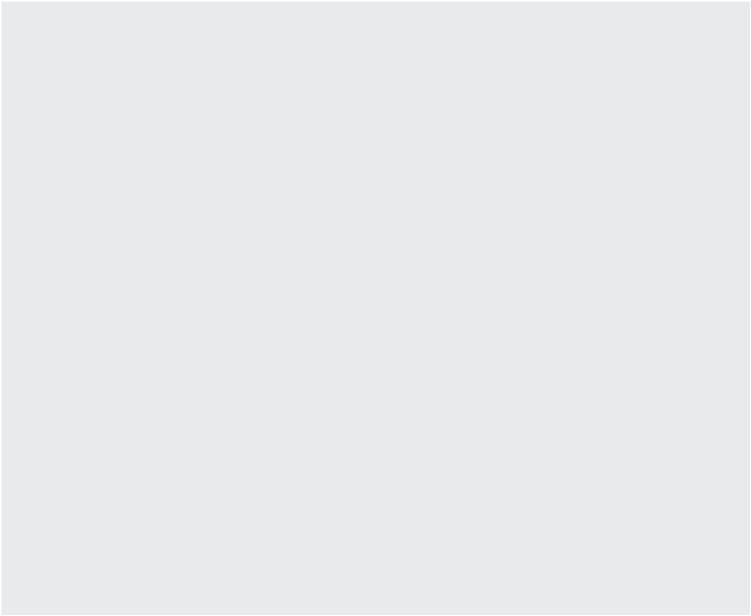


Prof. Marcelo Rovai

UNIFEI

Person Detection:
Application Architecture

✓



X

Person Detection using Transfer Learning Model

Code Walkthrough!

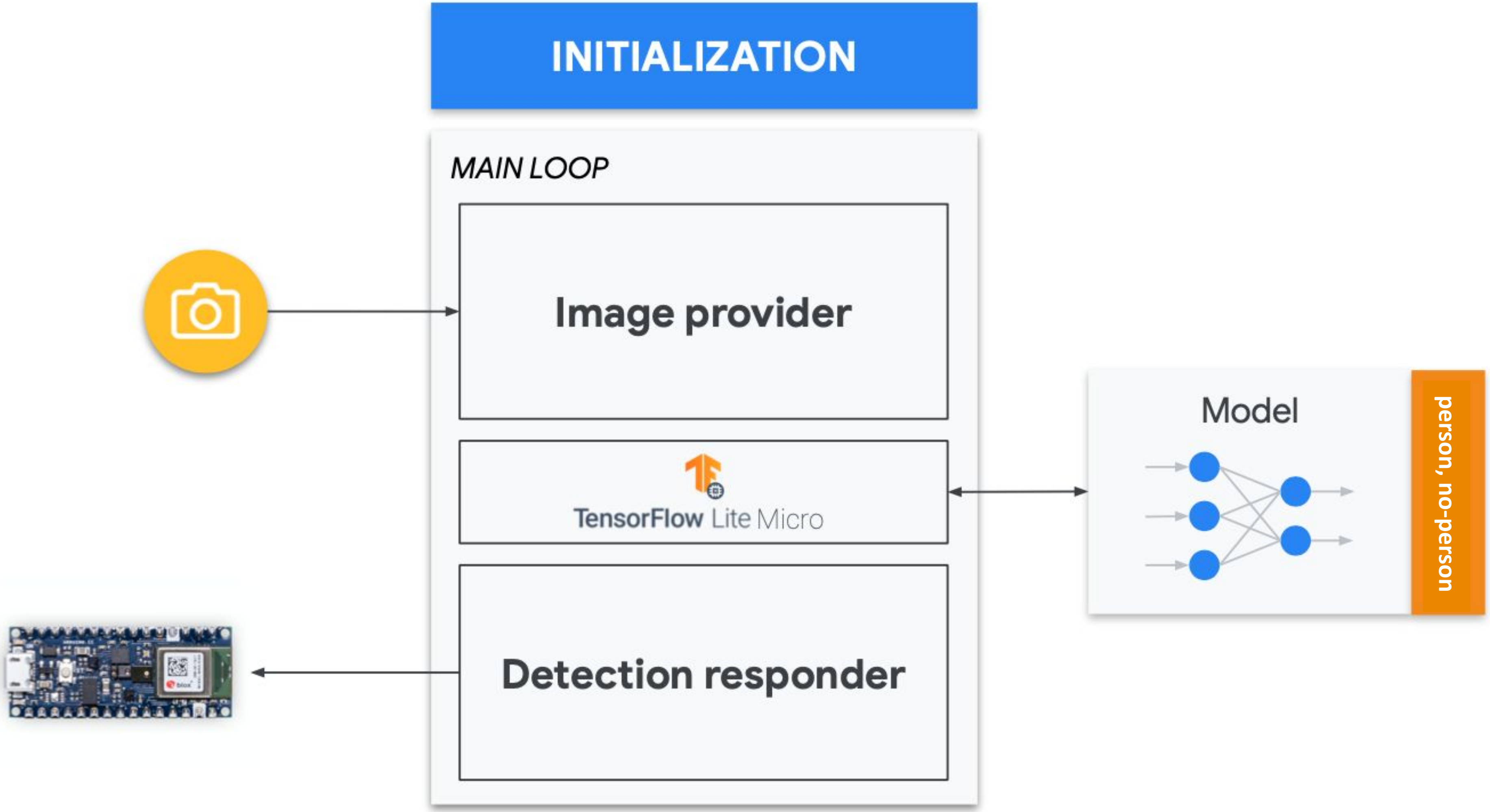
person_detection.ino (Arduino IDE TFLite Example)

TensorFlow Lite Micro

TensorFlow Lite Micro - Paper 🗎

MLSys 2021: TensorFlow Lite Micro TFLM ▶

Person Detection **Components**



INITIALIZATION

MAIN LOOP

Image provider

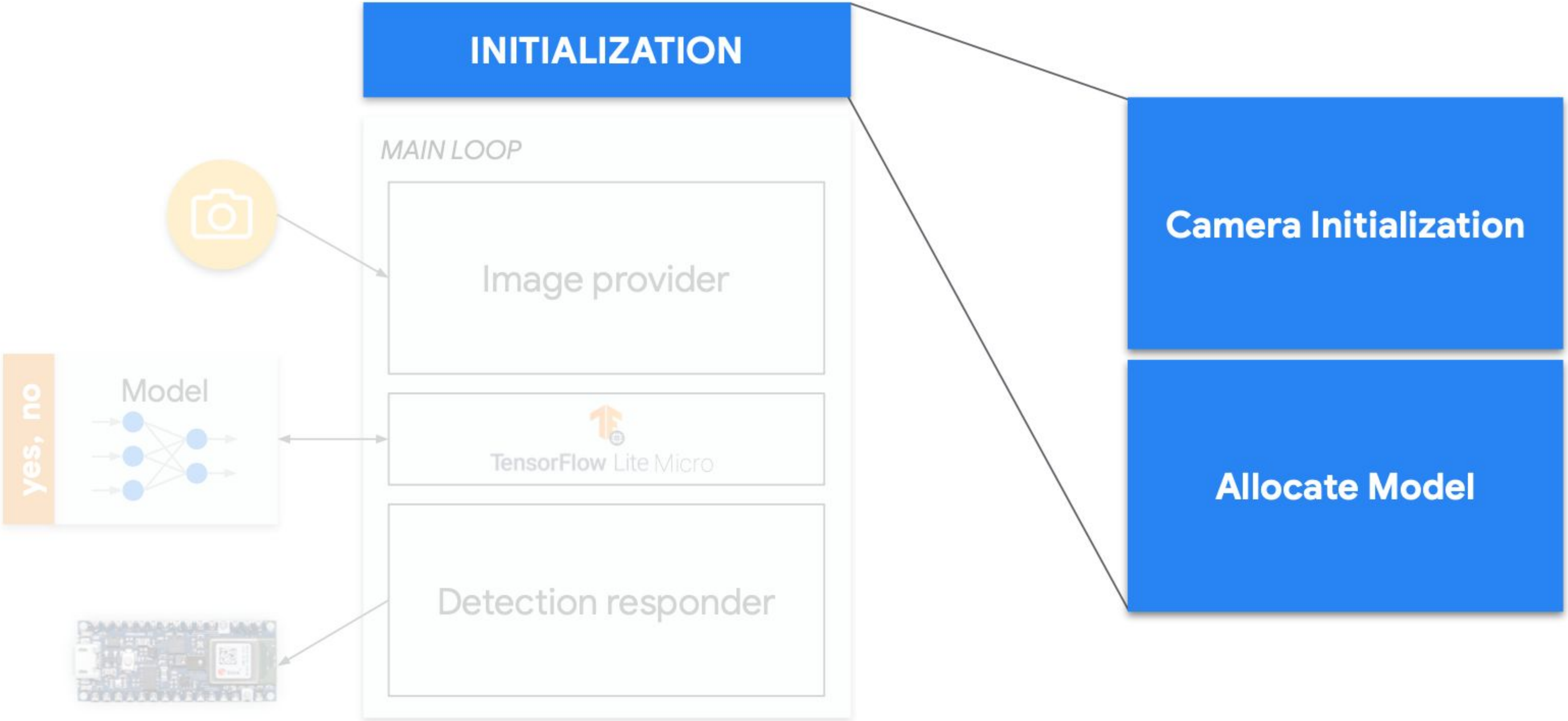
TensorFlow Lite Micro

Detection responder

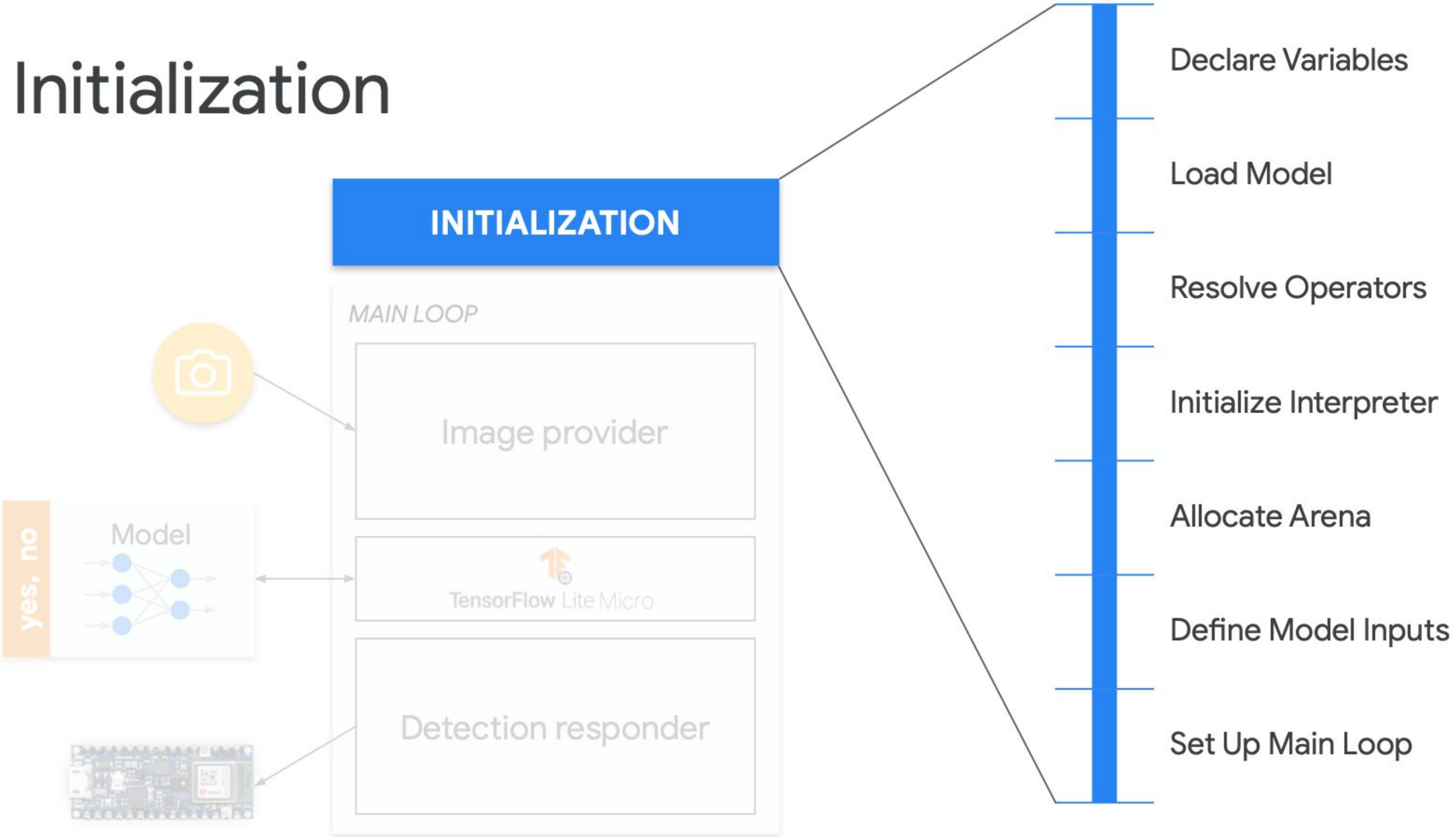
Model

person, no-person

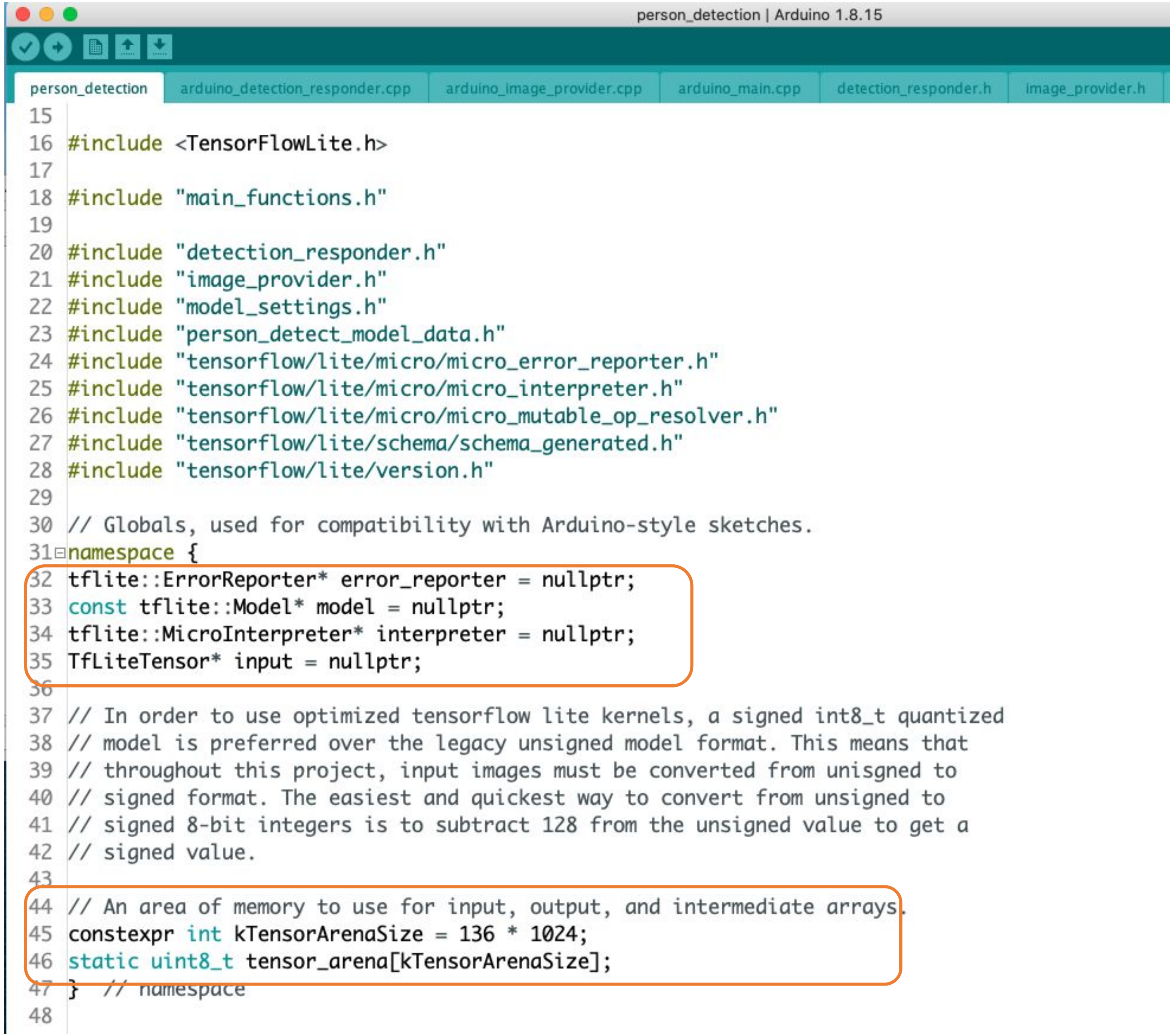
Initialization



Initialization



- Declare Variables
- Load Model
- Resolve Operators
- Initialize Interpreter
- Allocate Arena
- Define Model Inputs
- Set Up Main Loop


person_detection | Arduino 1.8.15
person_detection
arduino_detection_responder.cpp
arduino_image_provider.cpp
arduino_main.cpp
detection_responder.h
image_provider.h

```

#include <TensorFlowLite.h>

#include "main_functions.h"

#include "detection_responder.h"
#include "image_provider.h"
#include "model_settings.h"
#include "person_detect_model_data.h"
#include "tensorflow/lite/micro/micro_error_reporter.h"
#include "tensorflow/lite/micro/micro_interpreter.h"
#include "tensorflow/lite/micro/micro_mutable_op_resolver.h"
#include "tensorflow/lite/schema/schema_generated.h"
#include "tensorflow/lite/version.h"

// Globals, used for compatibility with Arduino-style sketches.
namespace {
tflite::ErrorReporter* error_reporter = nullptr;
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;
static uint8_t tensor_arena[kTensorArenaSize];
}  // namespace

```

Declare Variables

Load Model

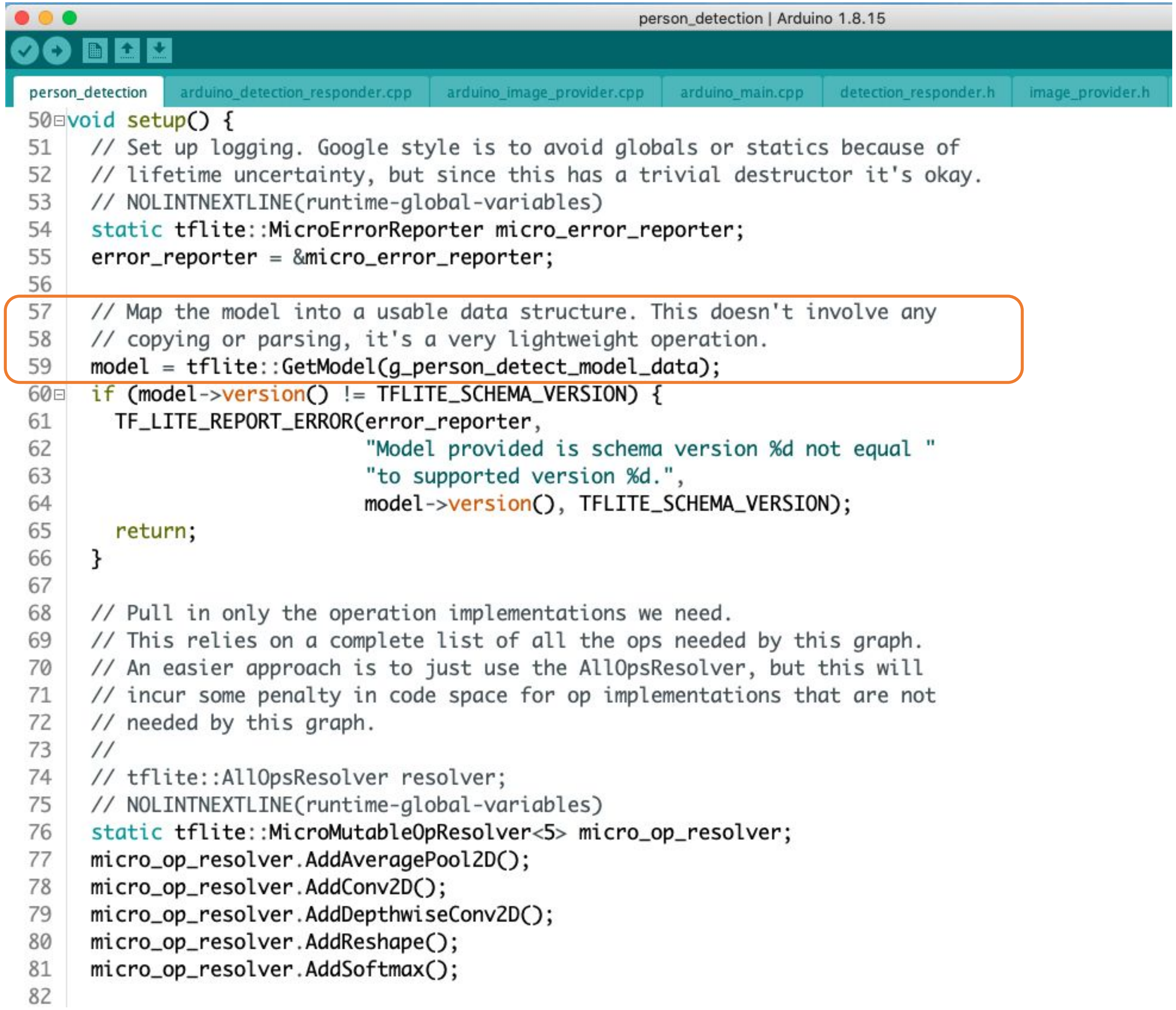
Resolve Operators

Initialize Interpreter

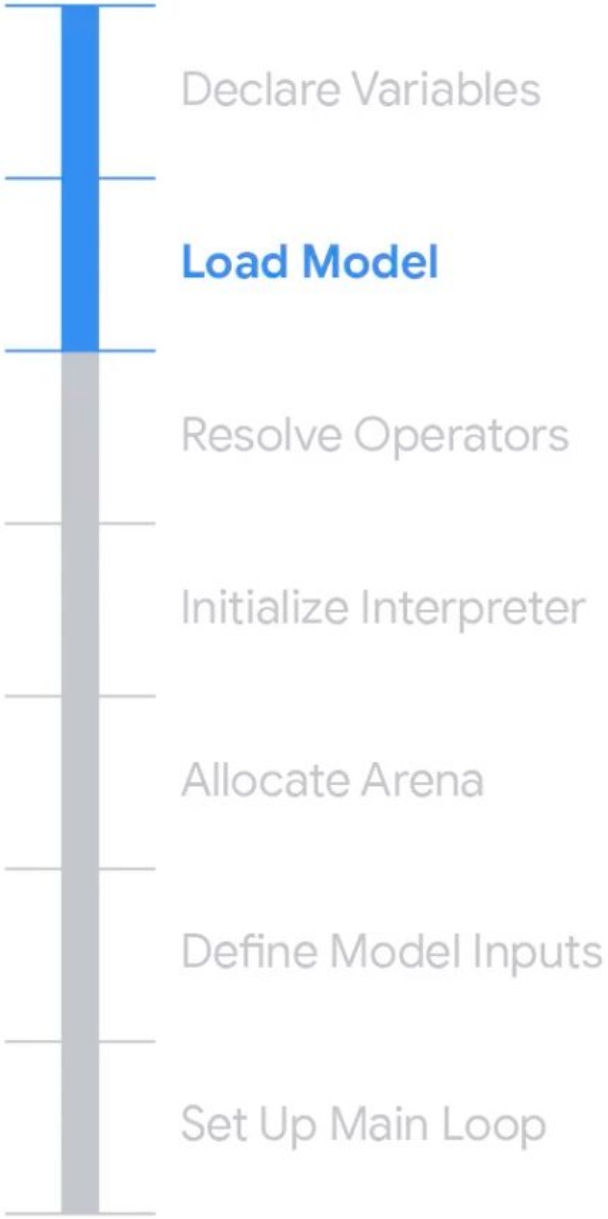
Allocate Arena

Define Model Inputs

Set Up Main Loop


person_detection | Arduino 1.8.15
person_detection
arduino_detection_responder.cpp
arduino_image_provider.cpp
arduino_main.cpp
detection_responder.h
image_provider.h

```
void setup() {
  // Set up logging. Google style is to avoid globals or statics because of
  // lifetime uncertainty, but since this has a trivial destructor it's okay.
  // NOLINTNEXTLINE(runtime-global-variables)
  static tflite::MicroErrorReporter micro_error_reporter;
  error_reporter = &micro_error_reporter;

  // 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) {
    TF_LITE_REPORT_ERROR(error_reporter,
                         "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

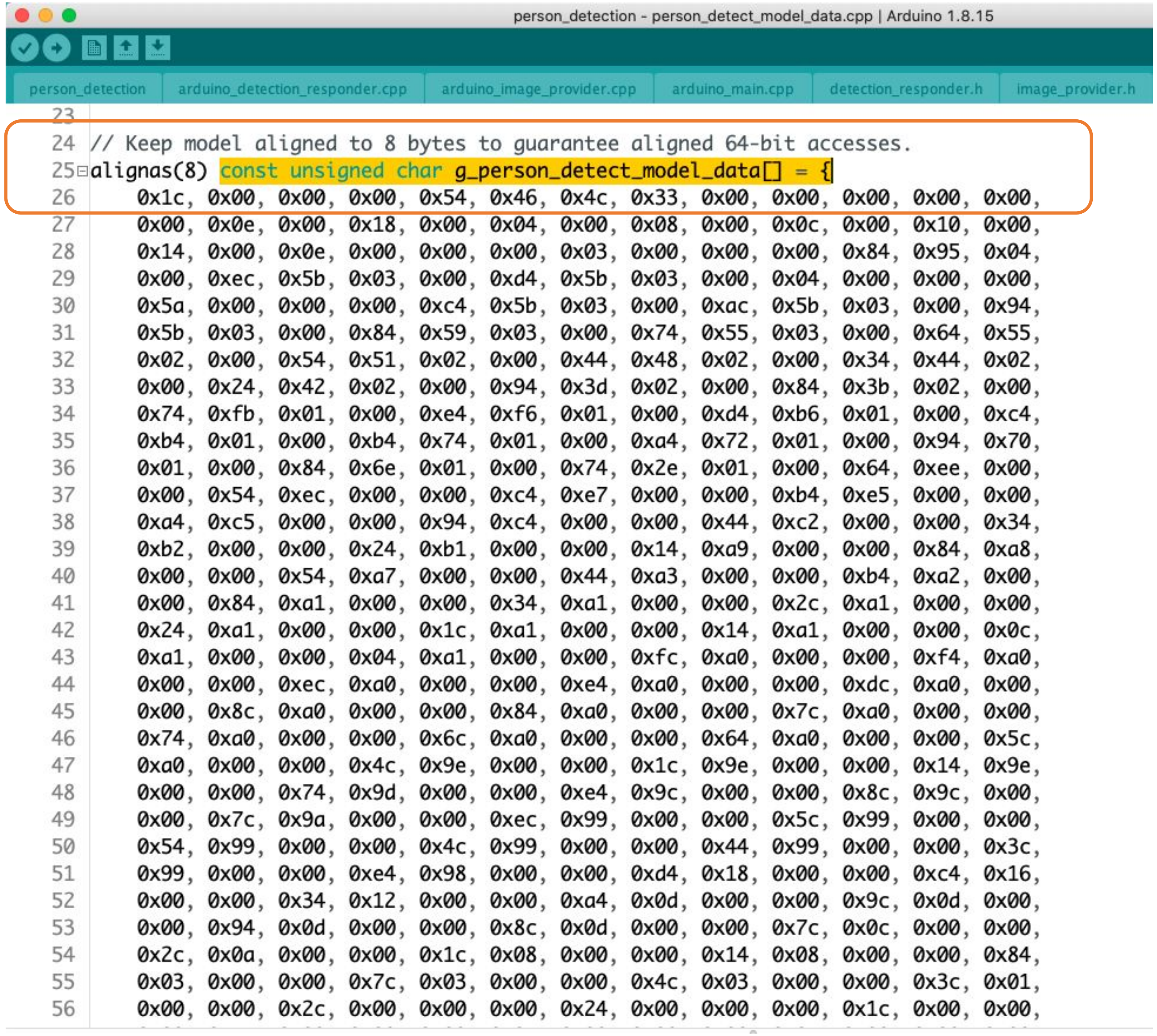
Resolve Operators

Initialize Interpreter

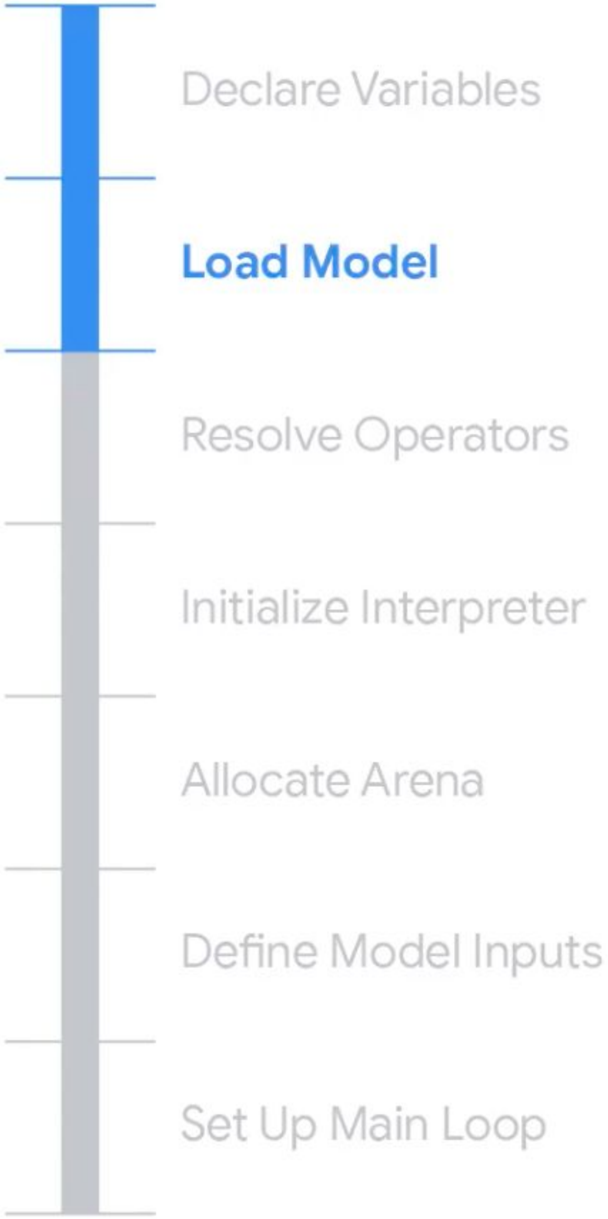
Allocate Arena

Define Model Inputs

Set Up Main Loop


person_detection - person_detect_model_data.cpp | Arduino 1.8.15
person_detection
arduino_detection_responder.cpp
arduino_image_provider.cpp
arduino_main.cpp
detection_responder.h
image_provider.h

```

// Keep model aligned to 8 bytes to guarantee aligned 64-bit accesses.
alignas(8) 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, 0xc4, 0x16,
    0x00, 0x00, 0x34, 0x12, 0x00, 0x00, 0xa4, 0x0d, 0x00, 0x00, 0x9c, 0x0d, 0x00,
    0x00, 0x94, 0x0d, 0x00, 0x00, 0x8c, 0x0d, 0x00, 0x00, 0x7c, 0x0c, 0x00, 0x00,
    0x2c, 0x0a, 0x00, 0x00, 0x1c, 0x08, 0x00, 0x00, 0x14, 0x08, 0x00, 0x00, 0x84,
    0x03, 0x00, 0x00, 0x7c, 0x03, 0x00, 0x00, 0x4c, 0x03, 0x00, 0x00, 0x3c, 0x01,
    0x00, 0x00, 0x2c, 0x00, 0x00, 0x00, 0x24, 0x00, 0x00, 0x00, 0x1c, 0x00, 0x00,
```



Declare Variables

Load Model

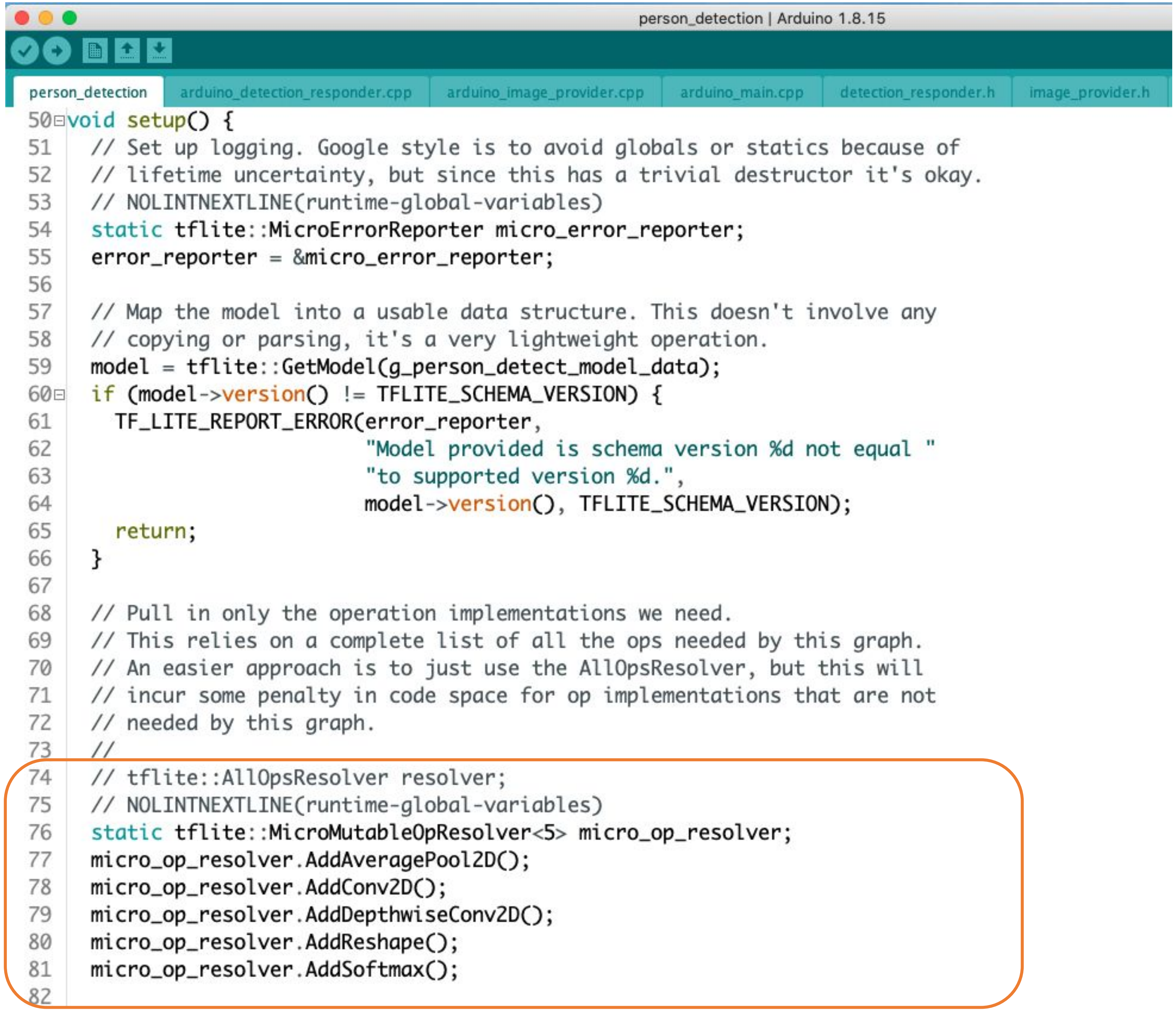
Resolve Operators

Initialize Interpreter

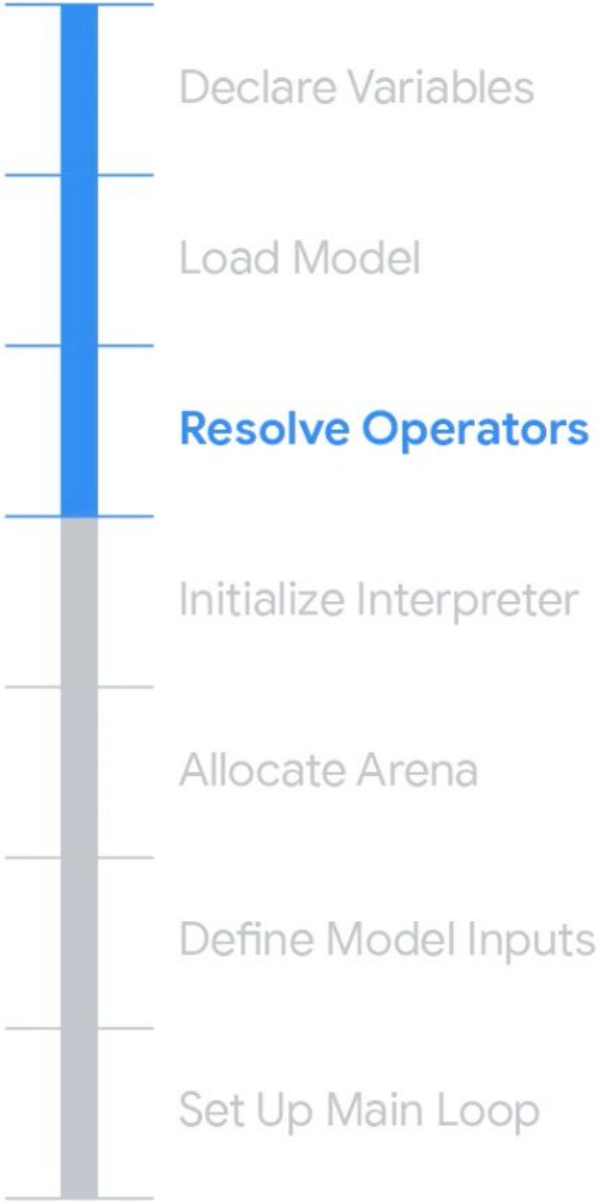
Allocate Arena

Define Model Inputs

Set Up Main Loop


person_detection | Arduino 1.8.15
person_detection
arduino_detection_responder.cpp
arduino_image_provider.cpp
arduino_main.cpp
detection_responder.h
image_provider.h

```
void setup() {
  // Set up logging. Google style is to avoid globals or statics because of
  // lifetime uncertainty, but since this has a trivial destructor it's okay.
  // NOLINTNEXTLINE(runtime-global-variables)
  static tflite::MicroErrorReporter micro_error_reporter;
  error_reporter = &micro_error_reporter;

  // 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) {
    TF_LITE_REPORT_ERROR(error_reporter,
                         "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

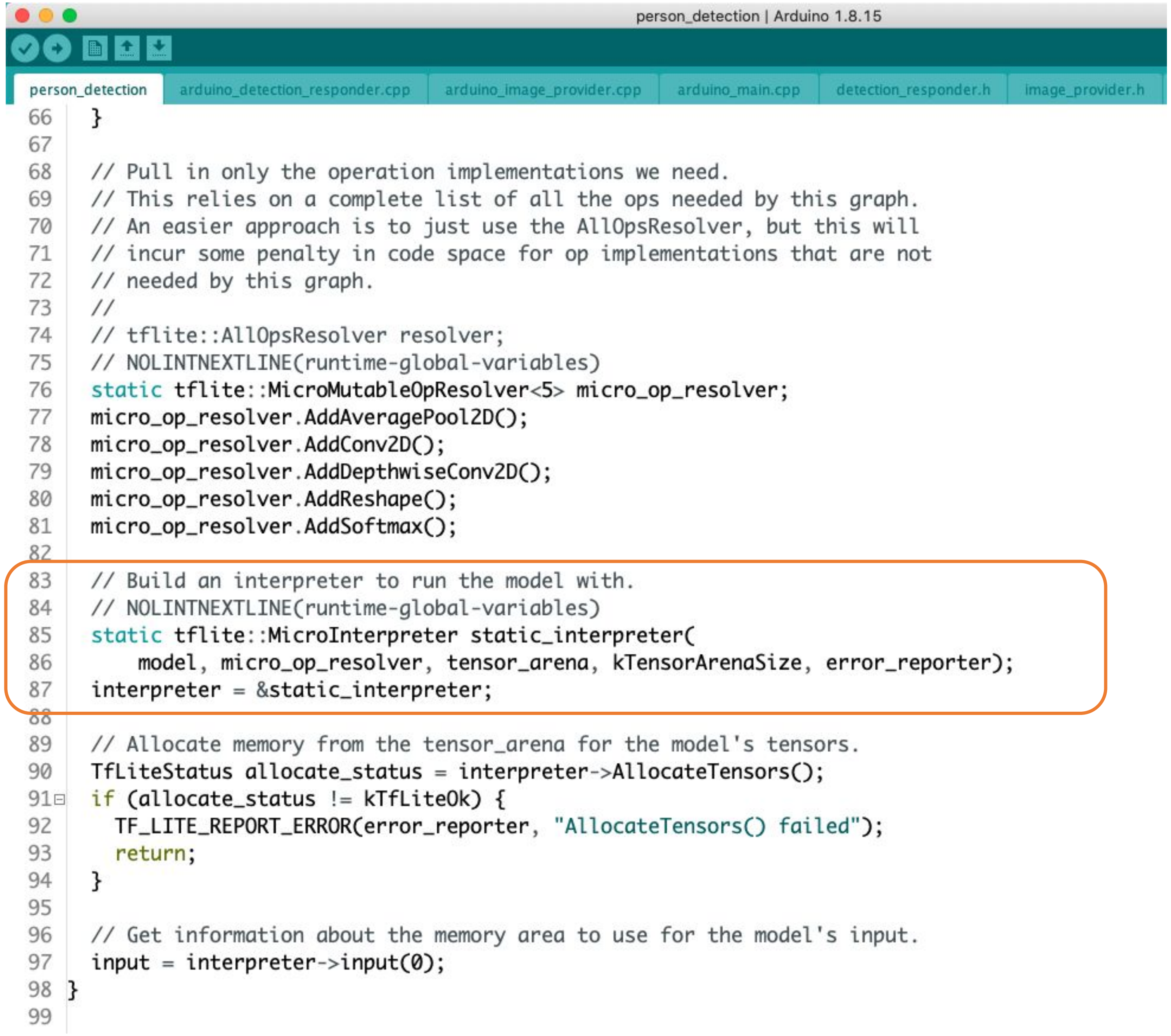
Resolve Operators

Initialize Interpreter

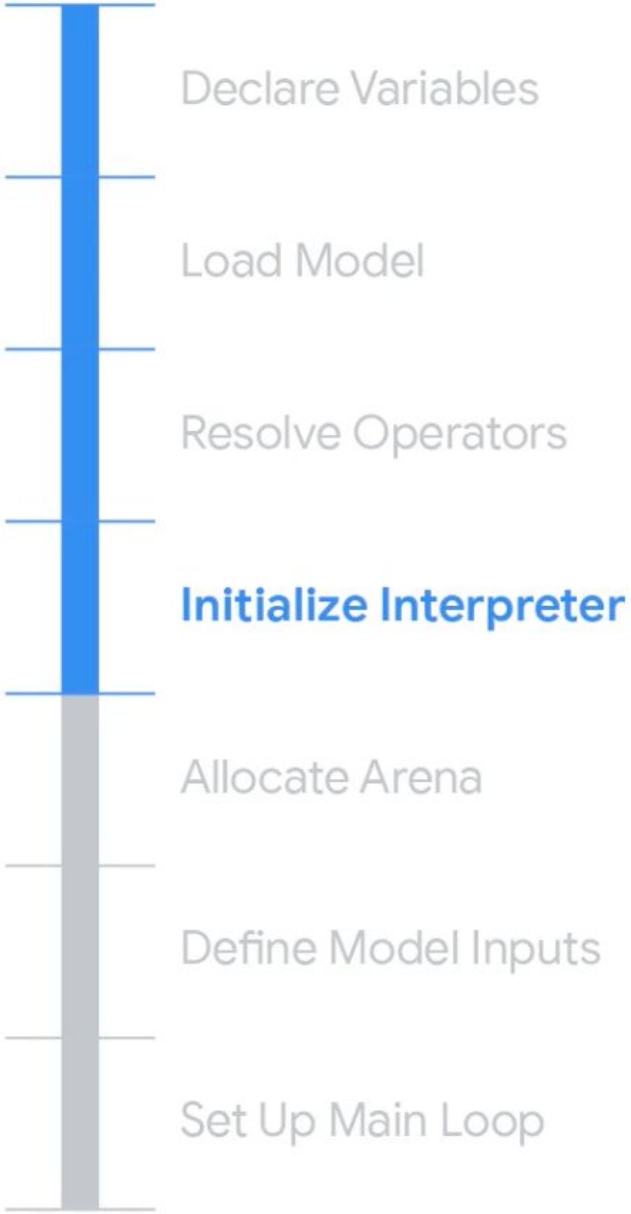
Allocate Arena

Define Model Inputs

Set Up Main Loop


person_detection | Arduino 1.8.15
person_detection
arduino_detection_responder.cpp
arduino_image_provider.cpp
arduino_main.cpp
detection_responder.h
image_provider.h

```
  }

  // 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();

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

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

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

```



Declare Variables

Load Model

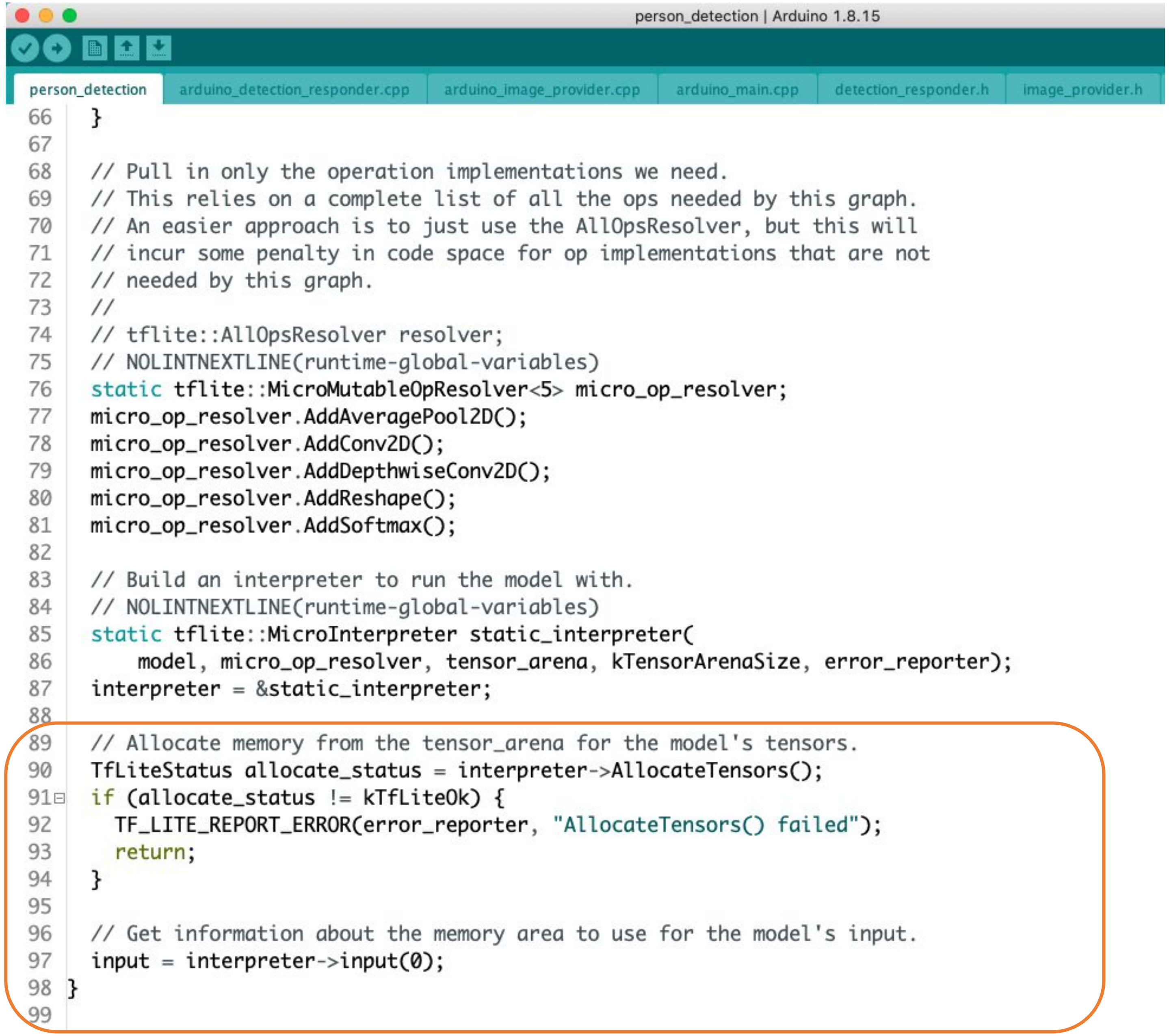
Resolve Operators

Initialize Interpreter

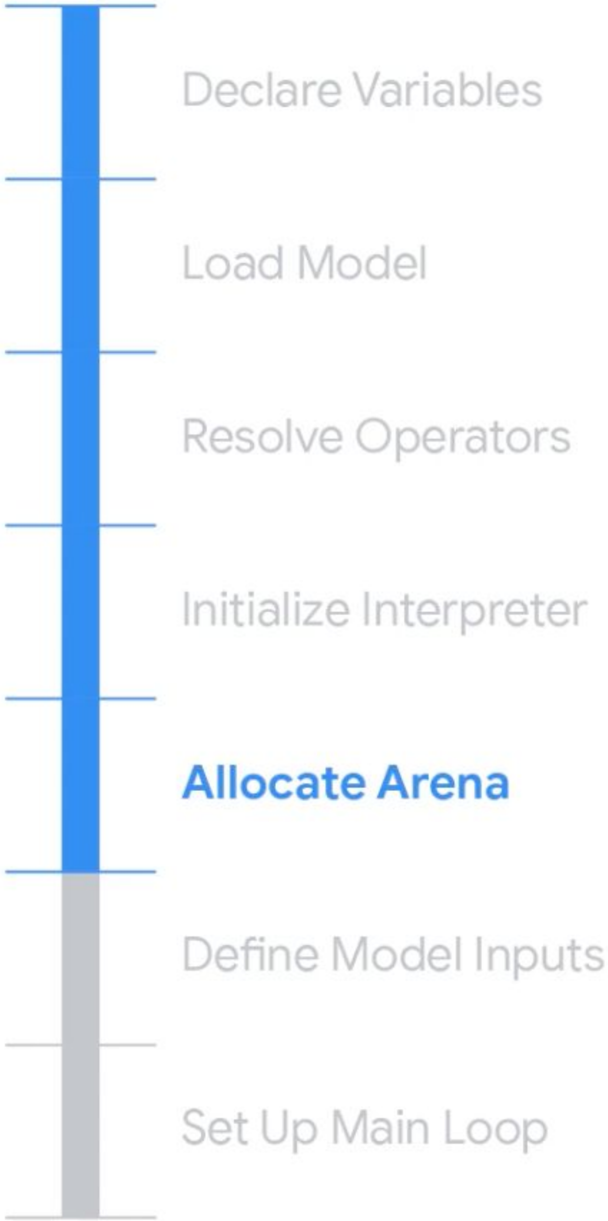
Allocate Arena

Define Model Inputs

Set Up Main Loop


person_detection | Arduino 1.8.15
person_detection
arduino_detection_responder.cpp
arduino_image_provider.cpp
arduino_main.cpp
detection_responder.h
image_provider.h

```
  }

  // 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();

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

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

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


Declare Variables

Load Model

Resolve Operators

Initialize Interpreter

Allocate Arena

Define Model Inputs

Set Up Main Loop

Initialization

Camera Initialization

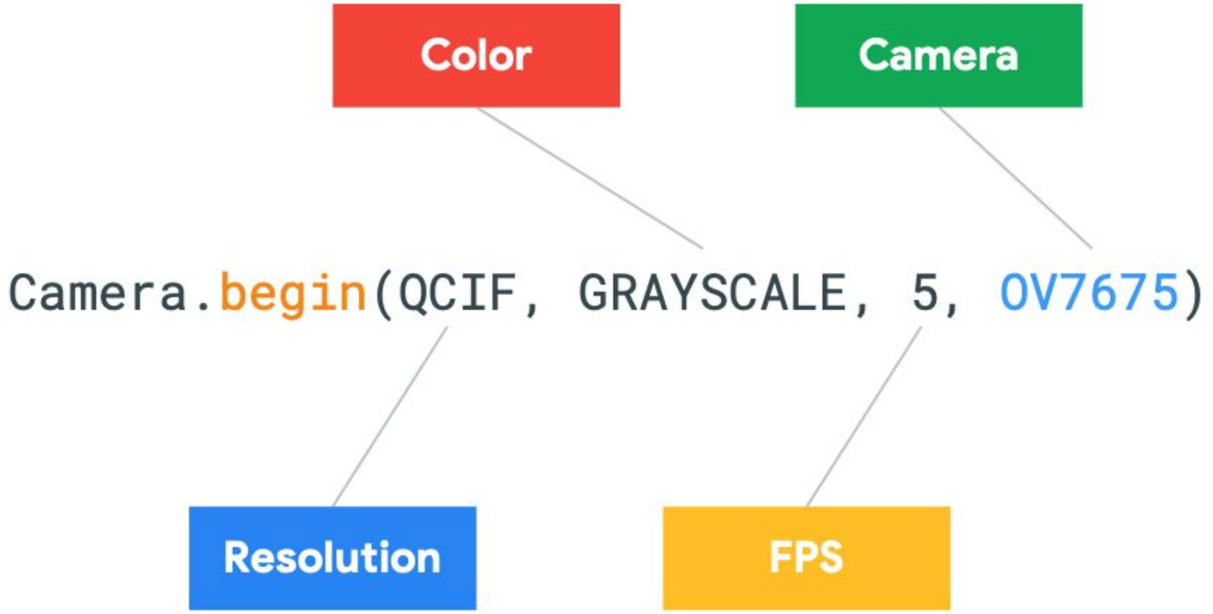
Allocate Model

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

Initialization

Camera Initialization

Allocate Model



Color

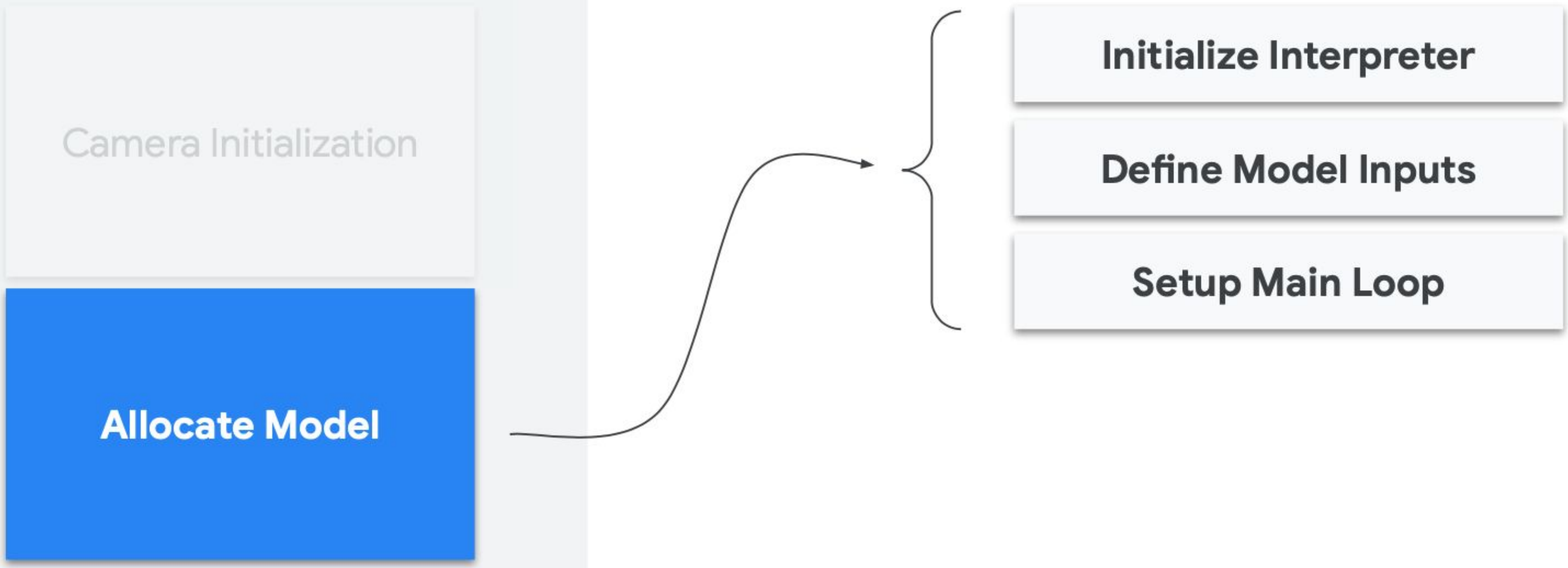
Camera

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

Resolution

FPS

Initialization



Camera Initialization

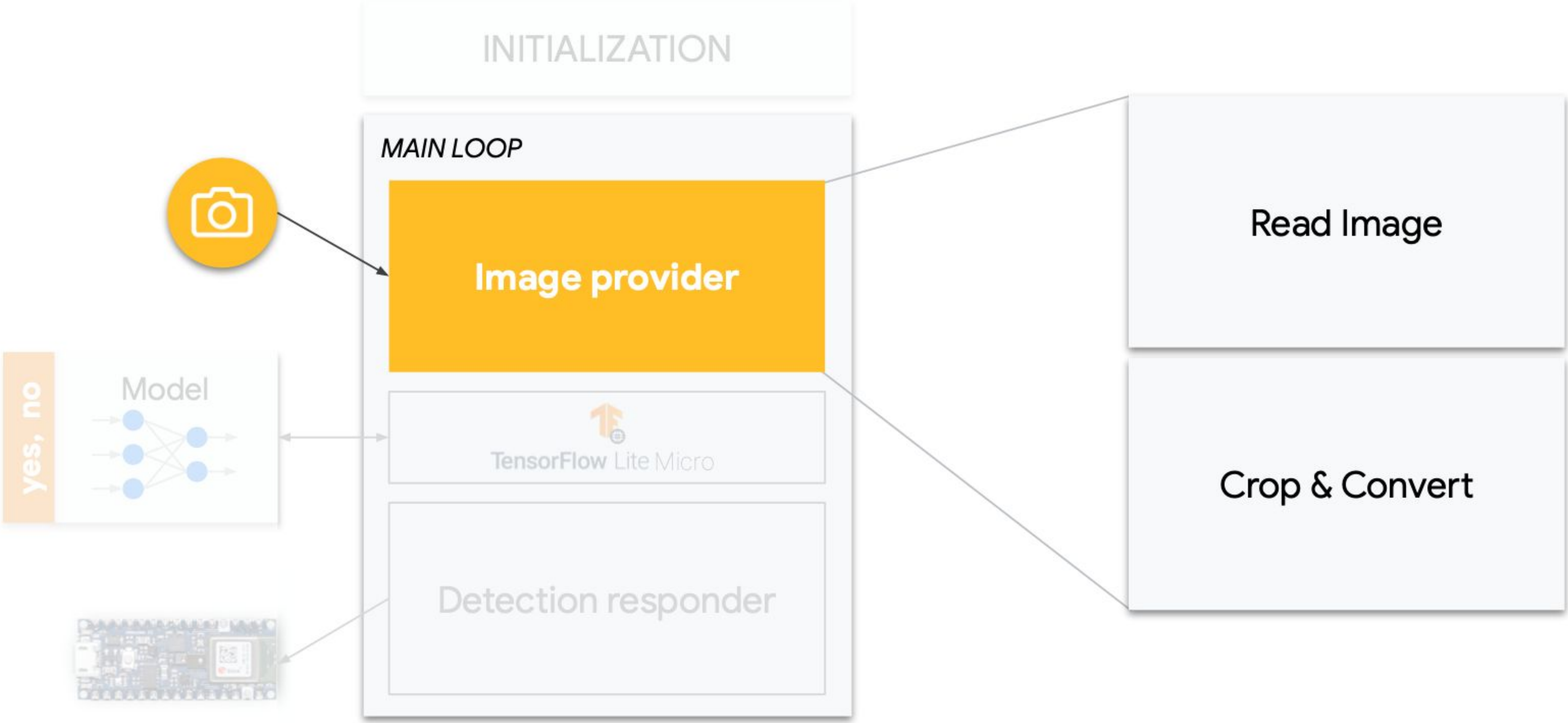
Allocate Model

Initialize Interpreter

Define Model Inputs

Setup Main Loop

Pre-processing



INITIALIZATION

MAIN LOOP

Image provider

TensorFlow Lite Micro

Detection responder

Model

yes, no

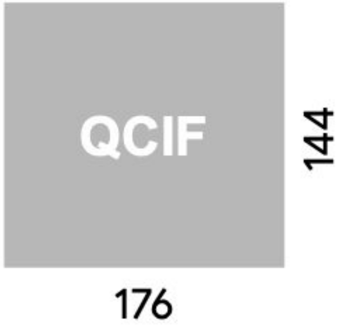
Read Image

Crop & Convert

Pre-processing

Read Image

Crop & Convert

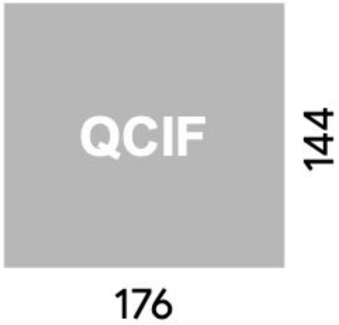


```
// Get an image from the camera module
TfLiteStatus GetImage(tflite::ErrorReporter* error_reporter,
          int image_width, int image_height, int channels,
          int8_t* image_data)
```

Pre-processing

Read Image

Crop & Convert

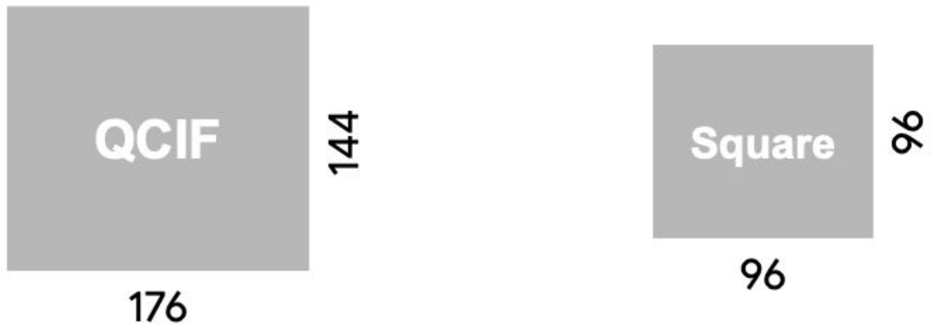


```
// Read camera data
Camera.readFrame(data);
```

Pre-processing

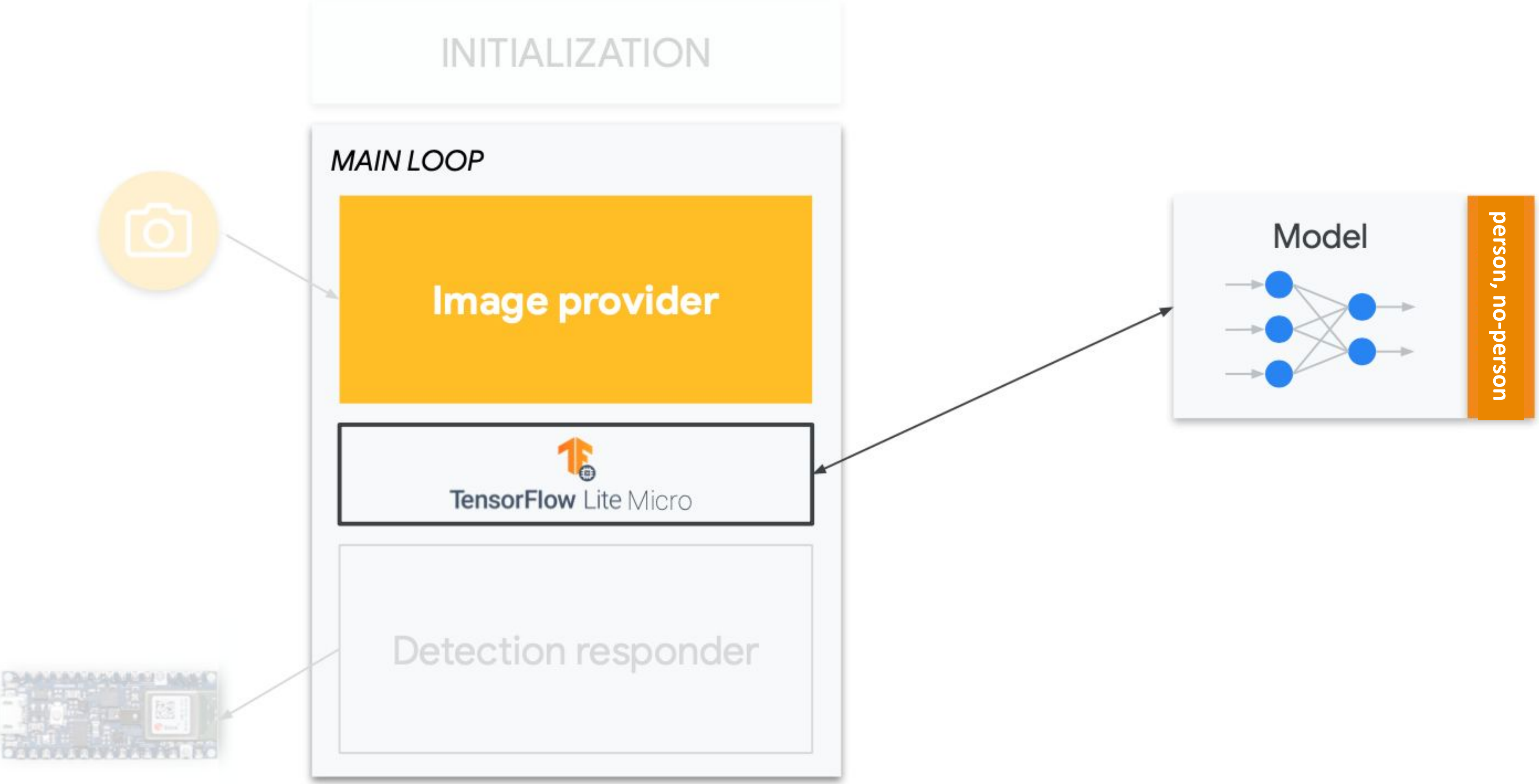
Read Image

Crop & Convert

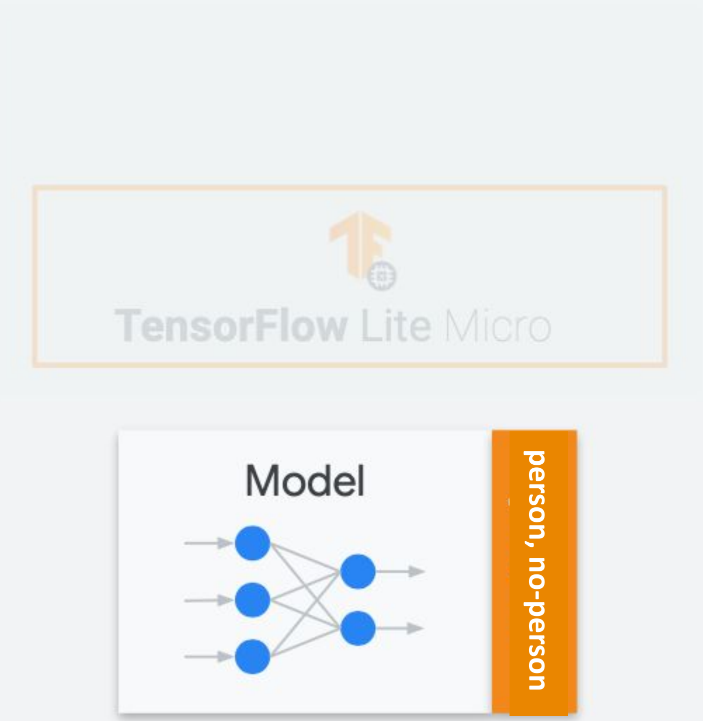


```
int min_x = (176 - 96) / 2;
int min_y = (144 - 96) / 2;
int index = 0;

// Crop 96x96 image. This lowers FOV, ideally we should downsample
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 input image to signed 8-bit
  }
}
```

Interpreter + Model

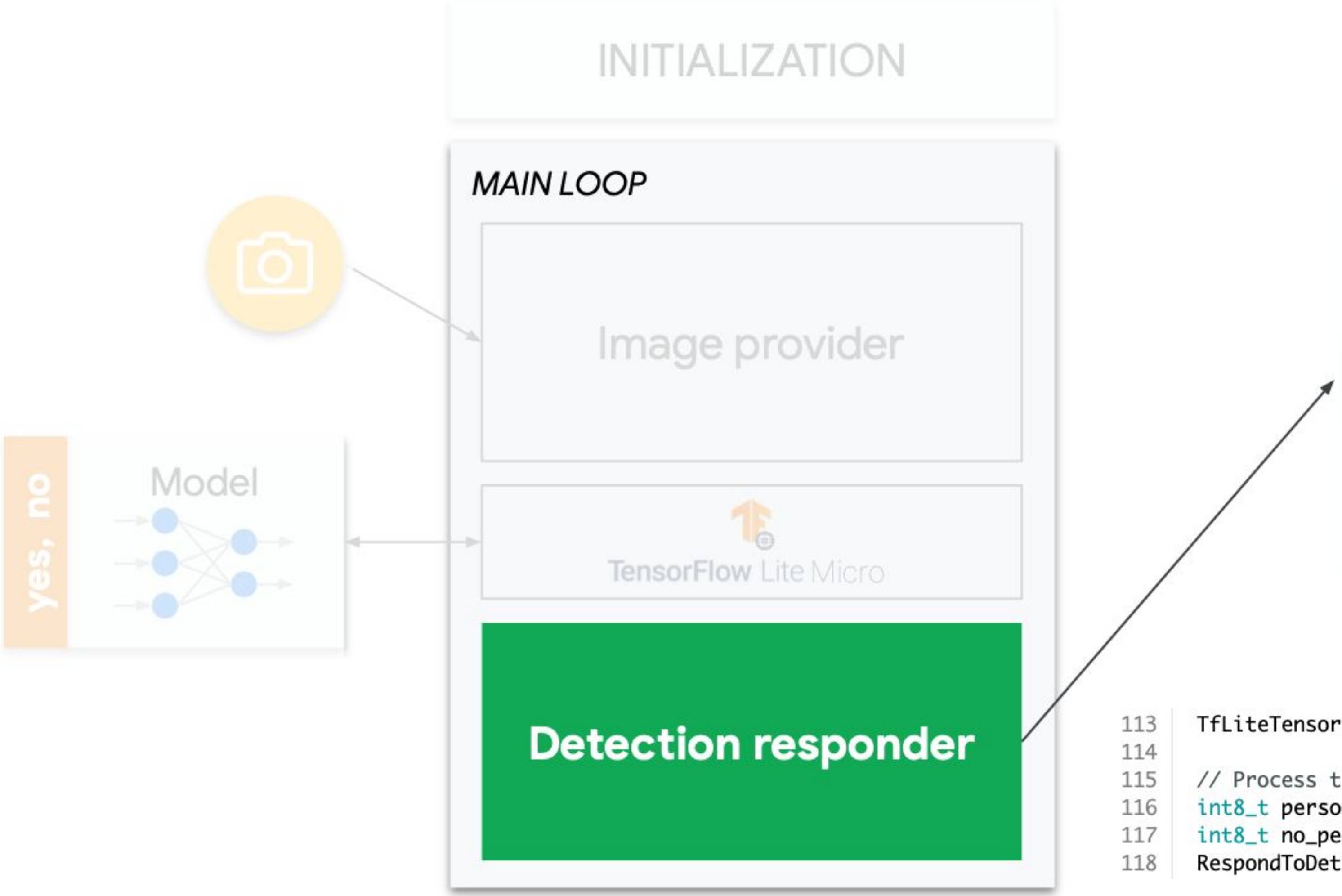


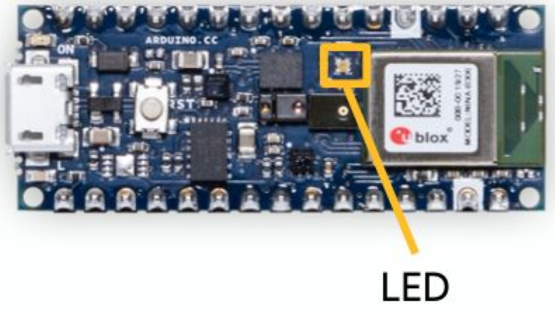
Interpreter + Model



```
kTfLiteOk != vww_interpreter->Invoke()
```

Post-processing





```
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];
RespondToDetection(error_reporter, person_score, no_person_score);
```

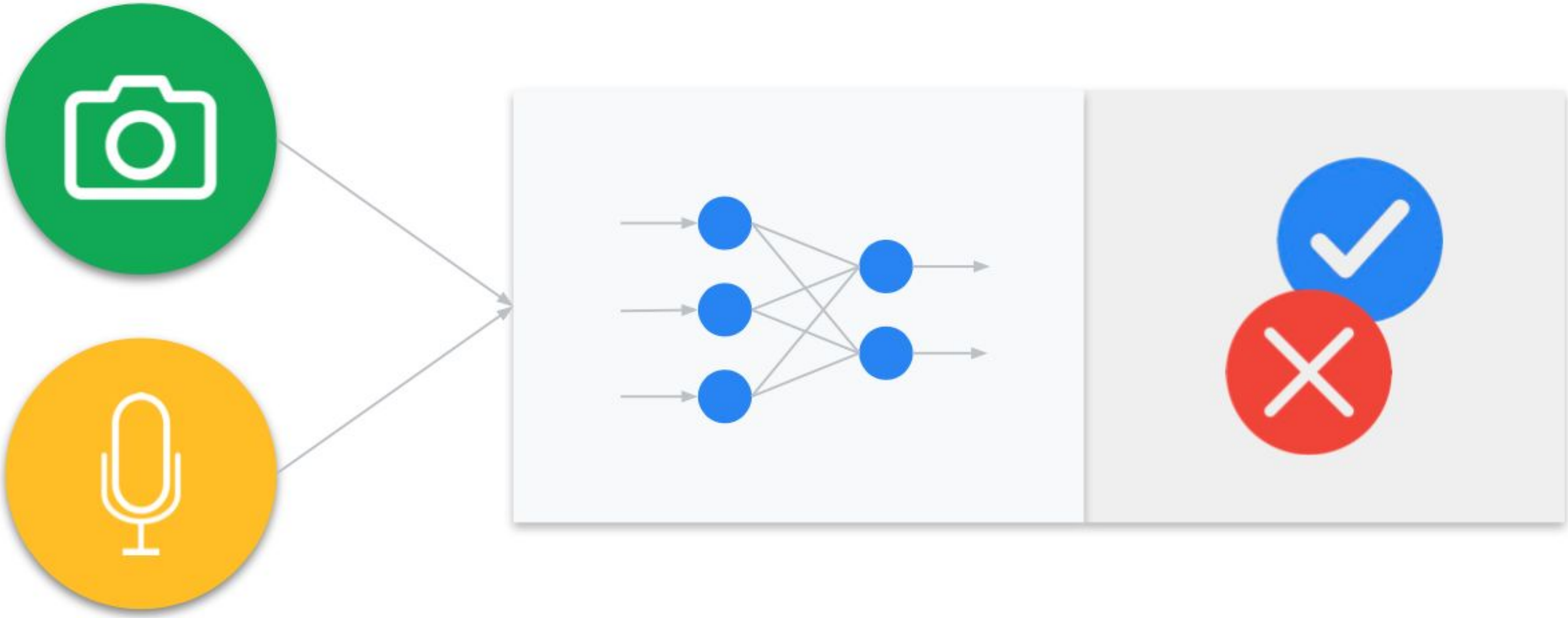
Detection responder

```
if (person_score > no_person_score) {
  digitalWrite(LEDG, LOW);
  digitalWrite(LEDR, HIGH);
} else {
  digitalWrite(LEDG, HIGH);
  digitalWrite(LEDR, LOW);
}
```

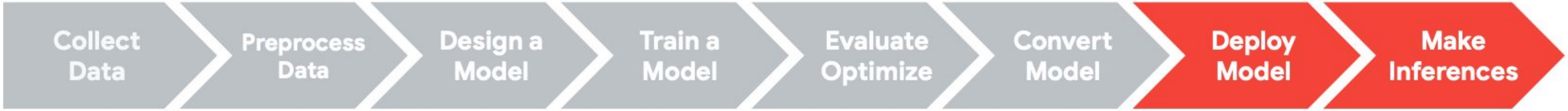
Person Detection:

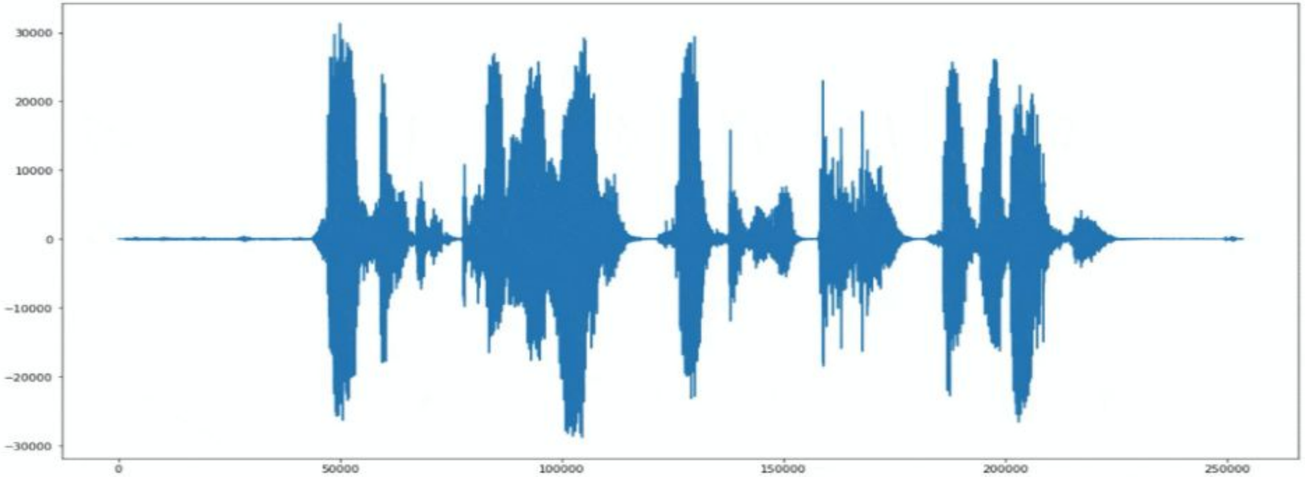
Multi-Tenancy

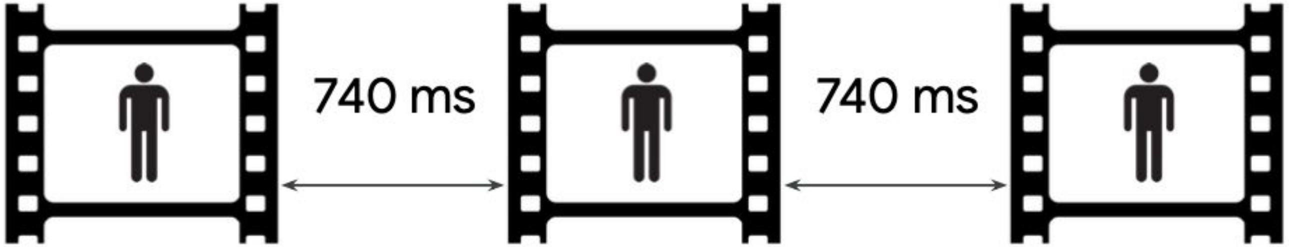
MultiModal



MultiModal ML Workflow





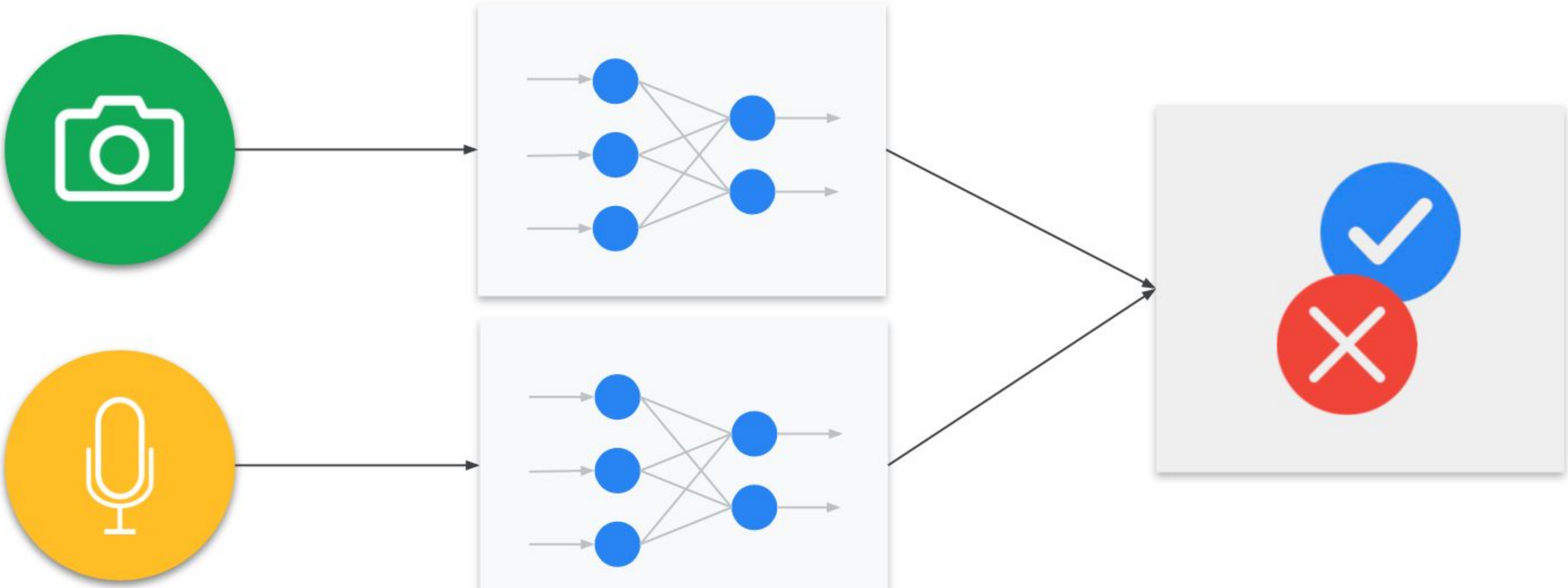


Example Person Detection Application

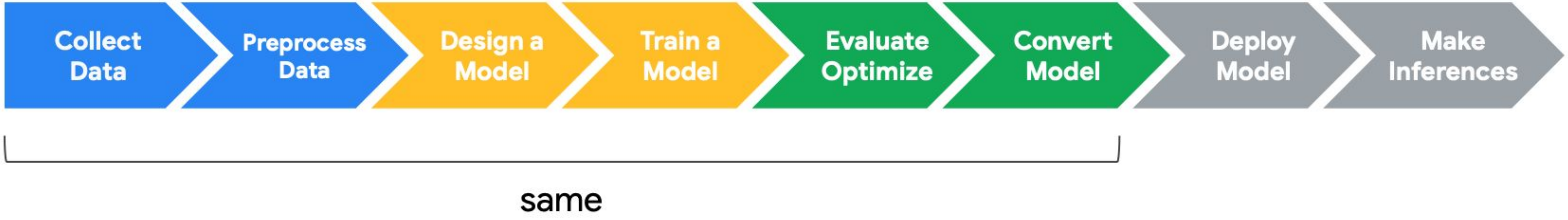
- Contact-free elevator control that enforces mask wearing

- Requires both **keyword spotting** and **mask detection**

MultiTenant



MultiTenant ML Workflow



Collect Data

Preprocess Data

Design a Model

Train a Model

Evaluate Optimize

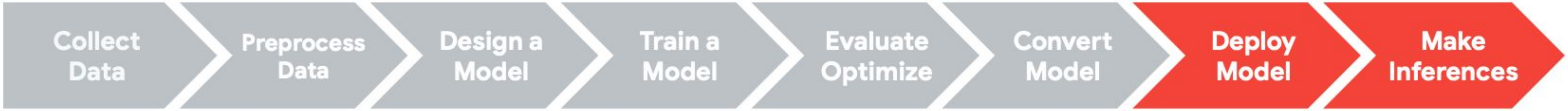
Convert Model

Deploy Model

Make Inferences

same

MultiTenant ML Workflow



Collect Data

Preprocess Data

Design a Model

Train a Model

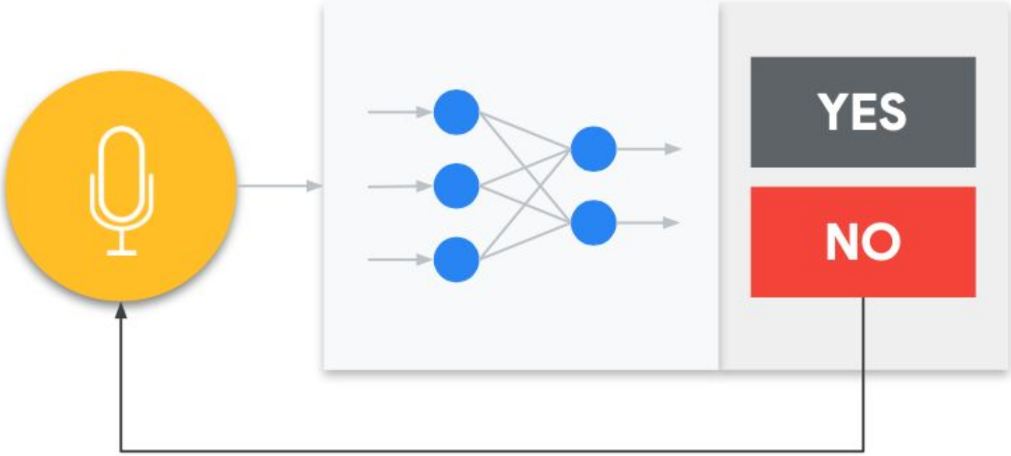
Evaluate Optimize

Convert Model

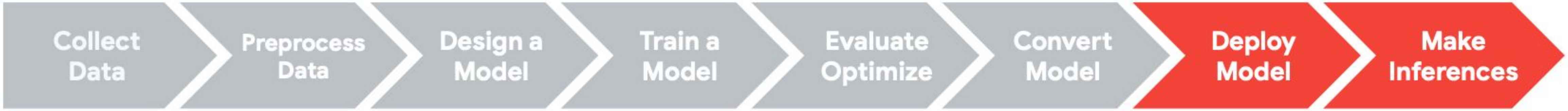
Deploy Model

Make Inferences

Cascade Multi Tenant



MultiTenant ML Workflow



Collect Data

Preprocess Data

Design a Model

Train a Model

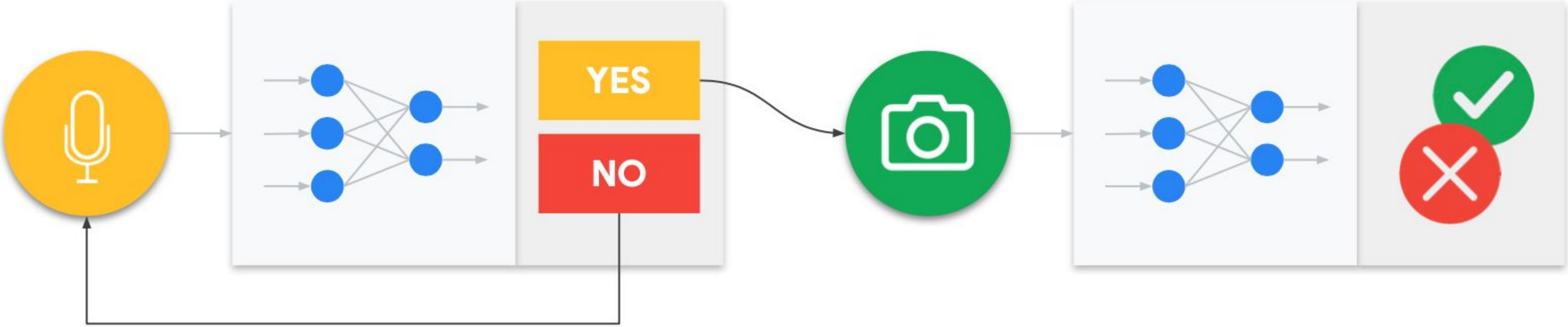
Evaluate Optimize

Convert Model

Deploy Model

Make Inferences

Cascade Multi Tenant



Reading Material

Main references

- Harvard School of Engineering and Applied Sciences - CS249r: Tiny Machine Learning
- Professional Certificate in Tiny Machine Learning (TinyML) – edX/Harvard
- Introduction to Embedded Machine Learning - Coursera/Edge Impulse
- Computer Vision with Embedded Machine Learning - Coursera/Edge Impulse
- Fundamentals textbook: "Deep Learning with Python" by François Chollet
- Applications & Deploy textbook: "TinyML" by Pete Warden, Daniel Situnayake
- Deploy textbook "TinyML Cookbook" by Gian Marco Iodice

I want to thank Shawn Hymel and Edge Impulse, Pete Warden and Laurence Moroney from Google, Professor Vijay Janapa Reddi and Brian Plancher from Harvard, and the rest of the TinyMLedu team for preparing the excellent material on TinyML that is the basis of this course at UNIFEI.

The IESTI01 course is part of the **TinyML4D**, an initiative to make TinyML education available to everyone globally.

Thanks

UNIFEI