# AIoT vision sensor with ESP32-S3
## TinyML implementation

Asst.Prof.Dr.Supachai Vorapojpisut
Thammasat University
vsupacha@engr.tu.ac.th


**Expected learning outcomes**

| Technical competencies | Activities |
|---|---|
| ML and TinyML concepts | Lectures: introduction to ML and TinyML<br>Demonstration: object detection device |
| Dev tools, SDK and API | Lectures: Python, TensorFlow API,<br>Demonstration: Edge Impulse workflow |
| Data collection and preparation | Lectures: data collection and labeling<br>Practice: image capture, label and upload |
| Model selection and training | Lectures: models for object detection<br>Practice: FOMO model training and tuning |
| ML inferencing | Lectures: Edge Impulse firmware<br>Practice: firmware customization |
| ML deploment | Lectures: MQTT protocol<br>Practice: capture, detect and report scenarios |

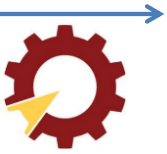
**Learning tools**

- Lilygo T-SIMCAM: ESP32S3 (16 MB Flash / 8MB PSRAM), OV2640 camera (2 Mpixels)
- Edge Impulse
- VS Code with Python and Platform.io extension + PySide

# 0. Edge computing – vision on MCU

https://www.youtube.com/watch?v=YmBWmXDLIdY

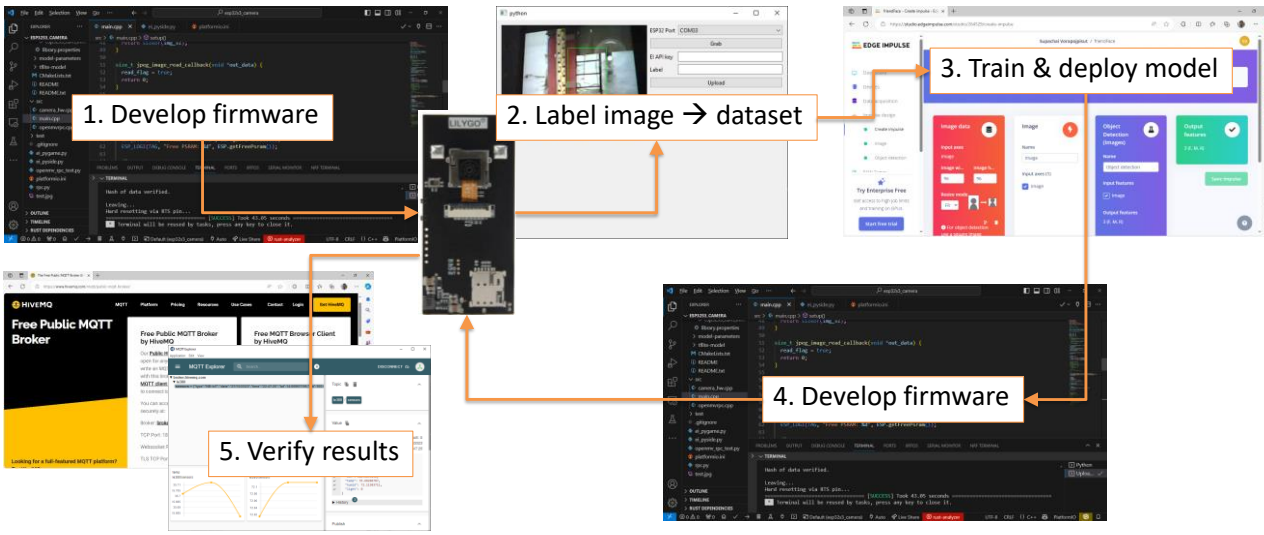
# 0. AIoT development workflow



VS Code + Platform.io extension

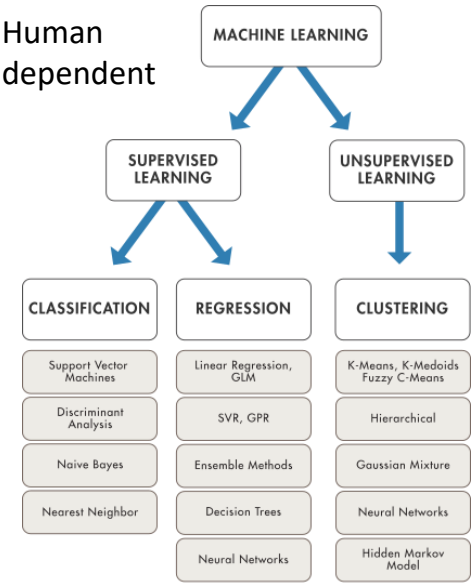
PySide6 + OpenCV

Edge impulse

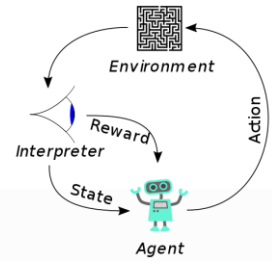


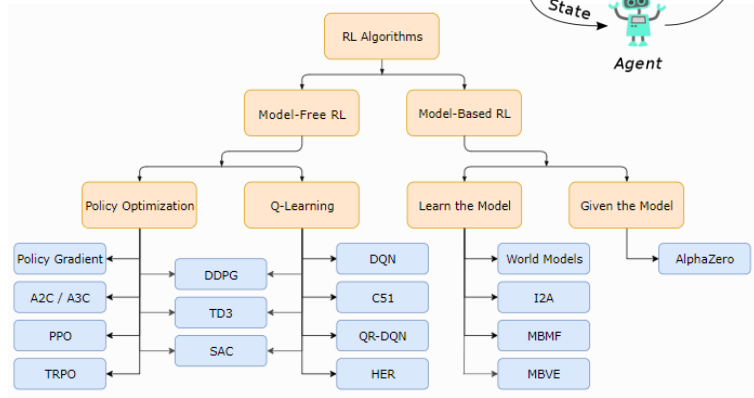
# 1. Machine learning

Human dependent




Human independent

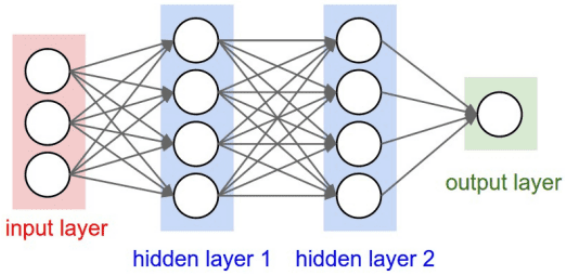


# 1. Neural network

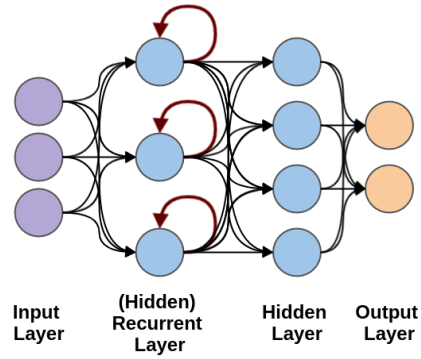
## Feedforward NN

- Features → input node
- Output types → output layer
- Complexity → hidden layer/nodes

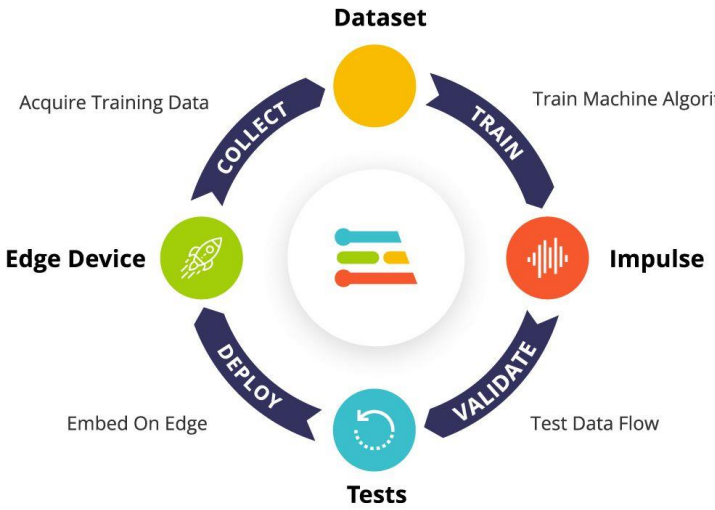



## Recurrent NN

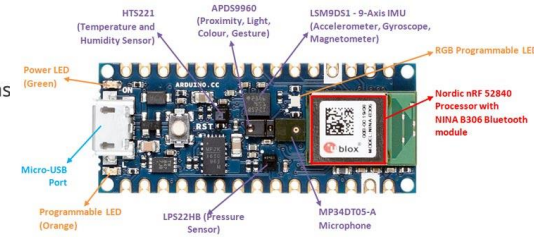
+ temporal/sequence → feedback loops

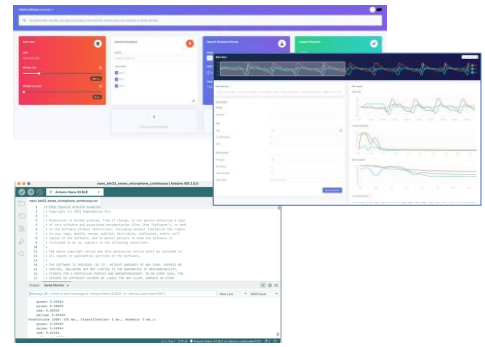


# 1. Edge Impulse workflow

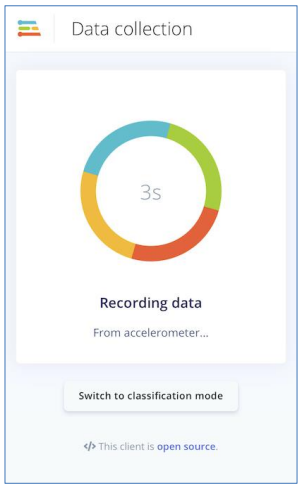


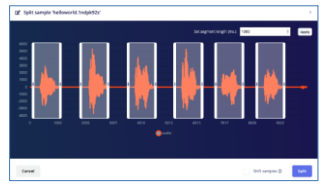

https://edgeimpulse.com/blog/getting-started-with-edge-impulse

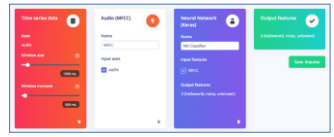


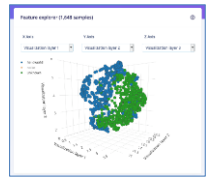



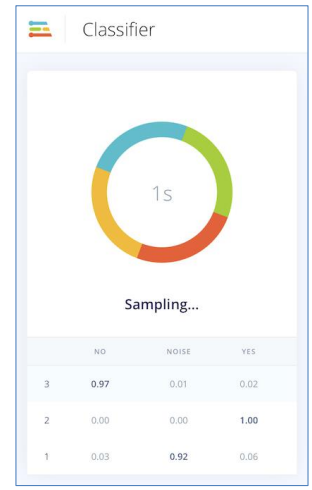
# 1. Smartphone as TinyML frontend



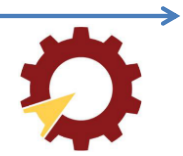





https://docs.edgeimpulse.com/docs/development-platforms/using-your-mobile-phone

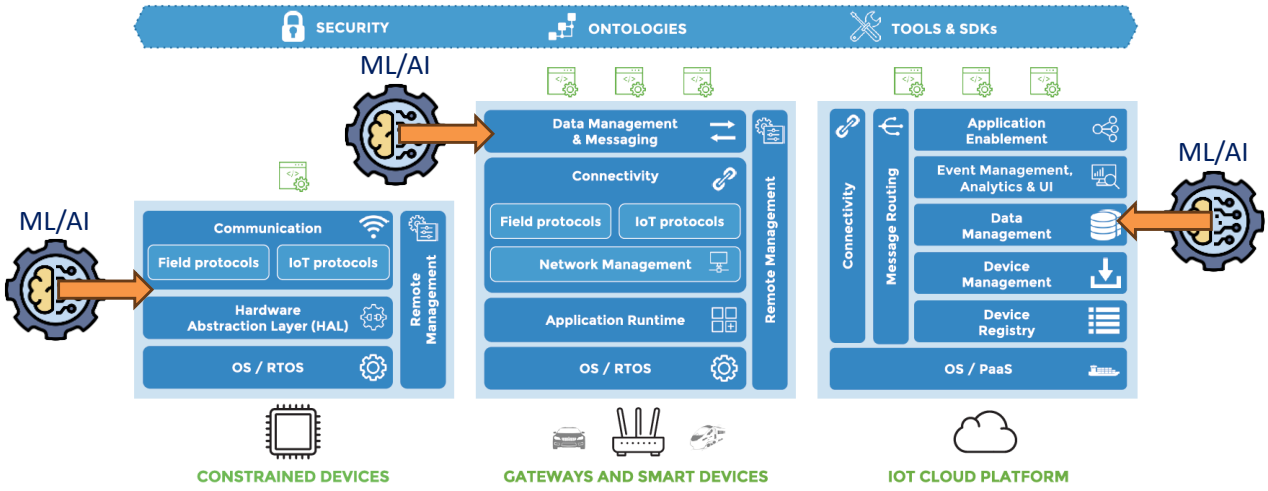
# Practice #1 edge impulse



1. Register and create Edge Impulse project
2. Activate smartphone for data collection
3. Collect voice dataset
4. Train classification model
5. Do live classification

https://edgeimpulse.com/university

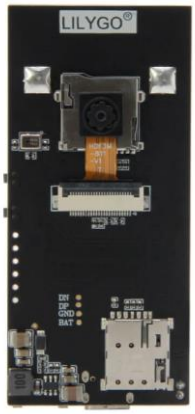
# 2. IoT software stack ← AI features

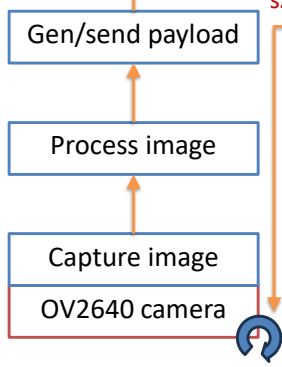



Ref: The Three Software Stacks Required for IoT Architectures

# 2. From edge vision to Internet

520 kB RAM
- 160 kB static variables
- 160 kB heap
- 200 kB IRAM




/tgr2023/ubon/water

{"node_id": 1, ... }

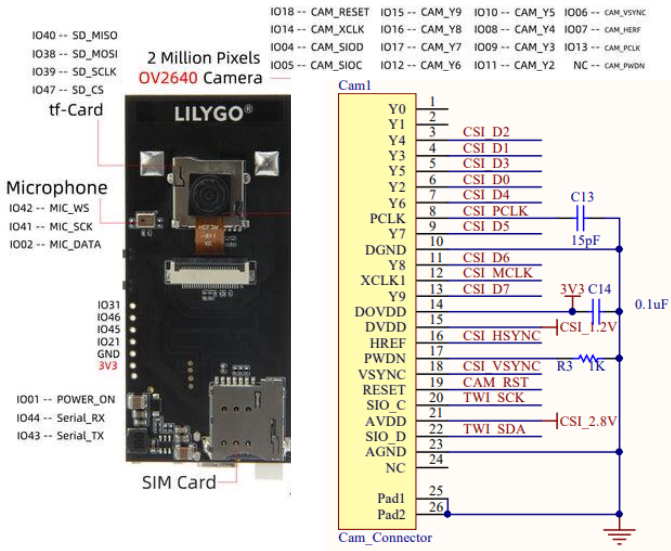
sz ≤ 256 bytes


| Format | Resolution | Data |
|---|---|---|
| UXGA | 1600 x 1200 | RAW<br>• RGB565<br>• YUV422<br>• YCbCr422<br>JPEG |
| | ... | |
| QVGA | 320 x 240 | |
| 240X240 | 240 x 240 | |
| | ... | |
| 96X96 | 96 x 96 | |

- Publisher-subscriber
  - Topic
  - Payload
- Bidirectional comm.
- QoS settings

https://www.hivemq.com/article/how-to-get-started-with-mqtt/

# 2. ESP32 camera interface




## esp32-camera library

- Need PSRAM

```
esp_camera_init(&camera_config);
```

- Pin configuration
- Pixel format: JPEG
- Frame size: 240x240
- Frame buffer: 2
- Mode: grab when empty

```
fb = esp_camera_fb_get();
```

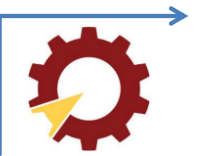
- Pointer to `camera_fb_t`

```
esp_camera_fb_return(fb);
```

- Return frame buffer

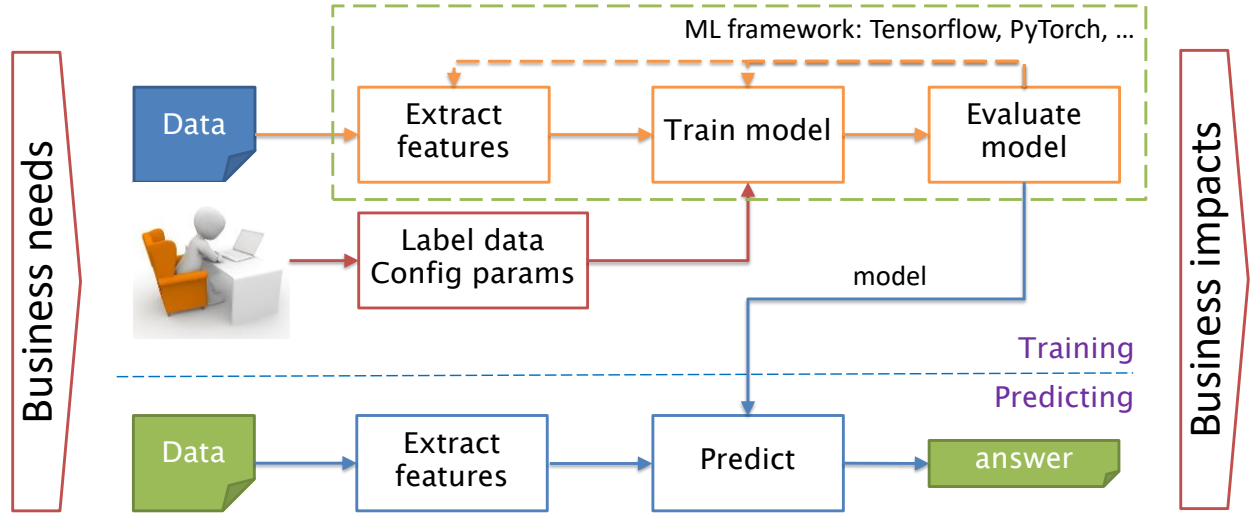
https://github.com/espressif/esp32-camera

# Practice #2 camera capture

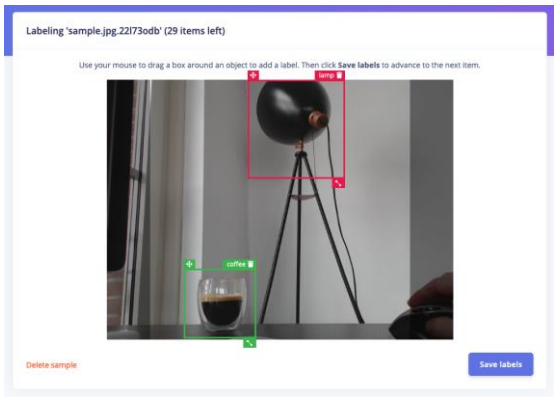


1. Open project tgr2023_05_camera
2. Write code to configure, initialize, and capture images on button pressed
   - Configure OV2640 pins
   - Initialize camera in `setup()`
   - Snapshot camera in `loop()`
3. Test when camera is covered and pointed to different scenes

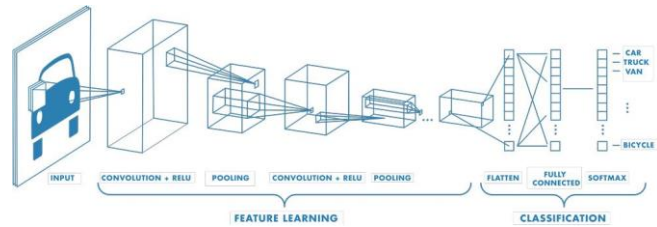
# 3. Supervised learning



# 4. Object detection




MobileNetV2 SSD FPN-Lite



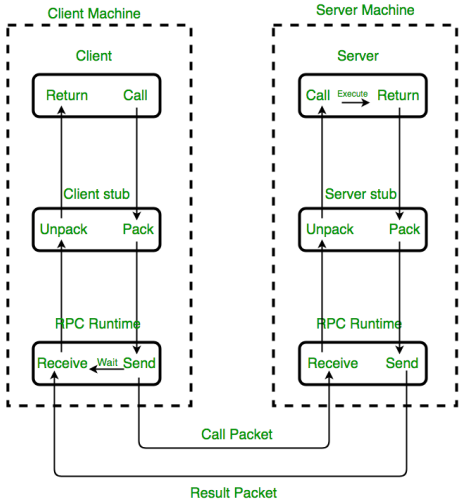

Model training

- Train/test dataset
- Image size
- Bounding box = location + size

https://docs.edgeimpulse.com/docs/tutorials/end-to-end-tutorials/object-detection/object-detection

https://docs.edgeimpulse.com/docs/tutorials/end-to-end-tutorials/object-detection/detect-objects-using-fomo

# 3. Remote Procedure Call (RPC) protocol




https://github.com/openmv/openmv-arduino-rpc

https://www.youtube.com/watch?v=WRHrqlKBZ3s

Remote device (server/slave)

- Register callback

```
rpc.register_callback("dummyFcn",
  dummyFcn)
```

- Prepare callback

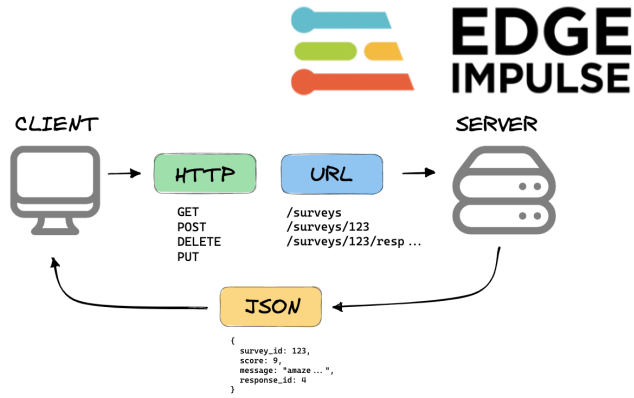
```
size_t dummyFcn(void *out){}
void dummyFcn(void *in,size_t len){}
```

Controller (client/master)

- Call remote task

```
rpc.call_no_args("dummyFcn",&in,
  sizeof(in))
rpc.call("dummyFcn", &out,
  sizeof(out))
```

# 3. Edge Impulse: ingestion API



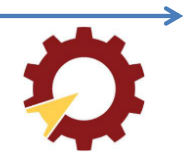

- API endpoint
  https://ingestion.edgeimpulse.com
  `POST /api/training/files`
  `POST /api/testing/files`
- Headers

| header | parameters |
|---|---|
| x-api-key | API key |
| x-label | label text |
| Content-type | multipart/form-data<br>application/json |

- Payload

https://edgeimpulse.readme.io/reference/ingestion-api

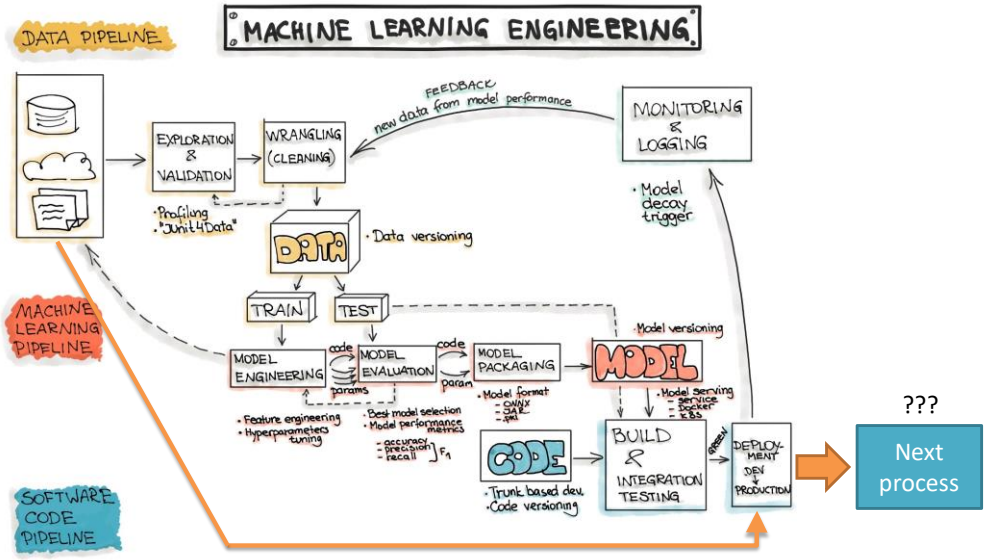
# Practice #3 AIoT data collection



1. Open project tgr2023_06_imglabel
2. Install Python and required libraries
   `pip install -r requirements.txt`
3. Modify main.cpp to add RPC callback
   - `button_read_callback()` read and return button status
   - `jpeg_image_snapshot_callback()` snapshot image
   - `jpeg_image_read_callback()` read image via Serial
4. Run ei_pyside.py to grab and label image
   - Use API key from Edge Impulse web

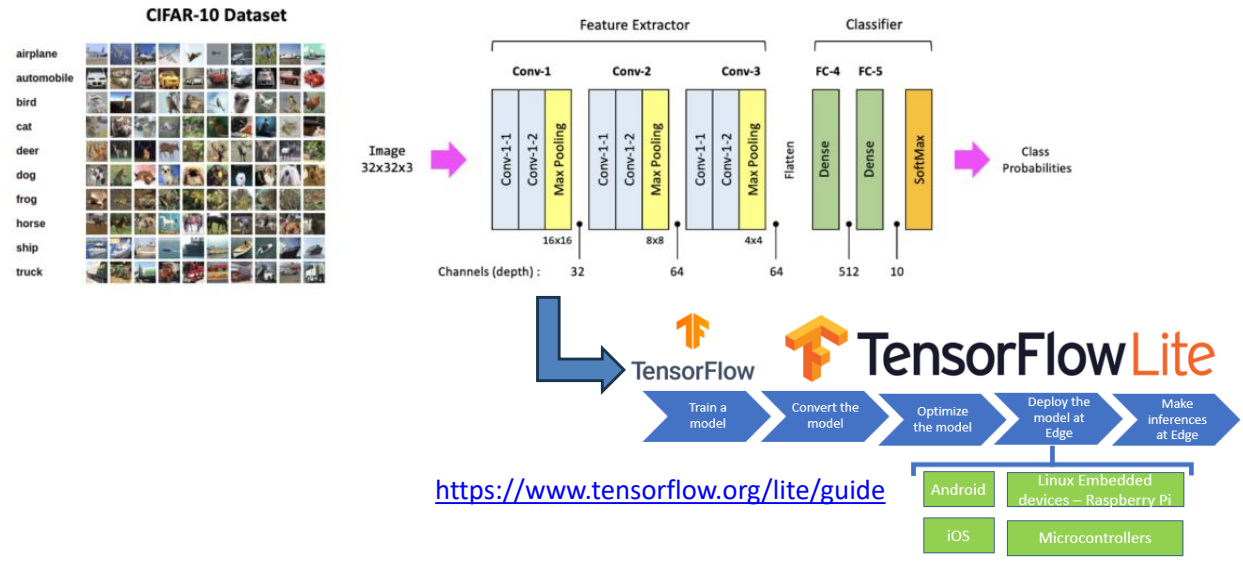
# 3. End-to-End ML workflow

- Human (staff, customer)
- Machine
- Other sources




https://ml-ops.org/content/end-to-end-ml-workflow

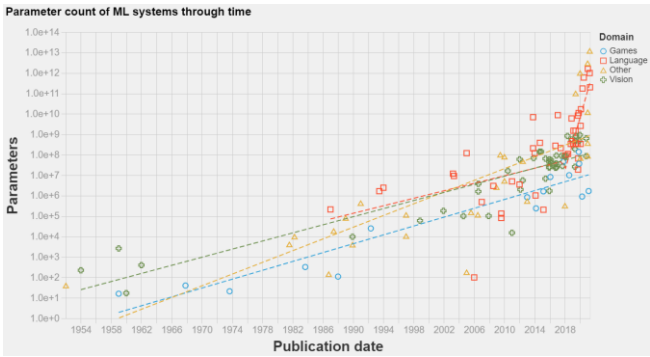
# 4. TensorFlow Lite

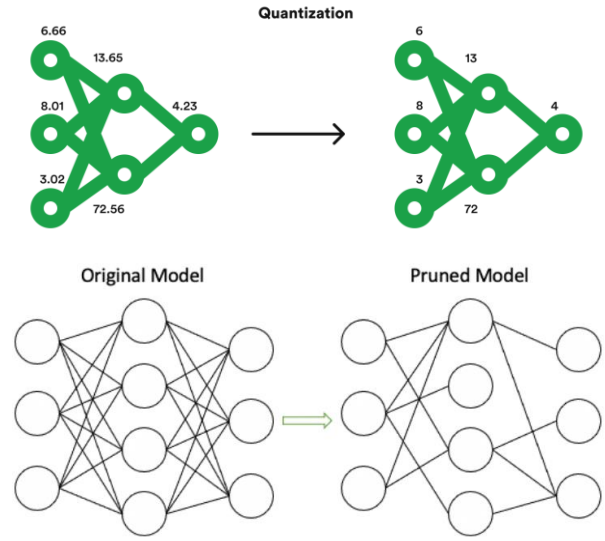



https://www.tensorflow.org/lite/guide

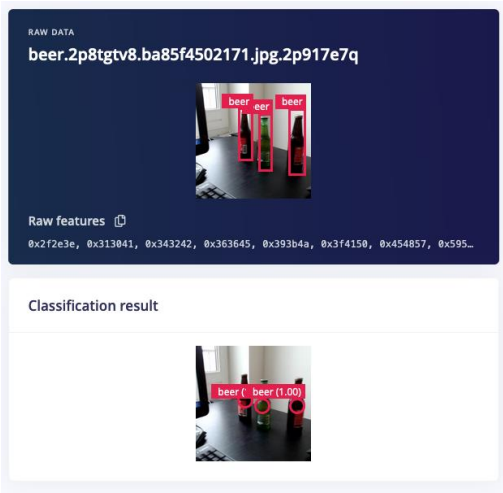
# 4. TinyML: quantization and pruning

Neural network parameters



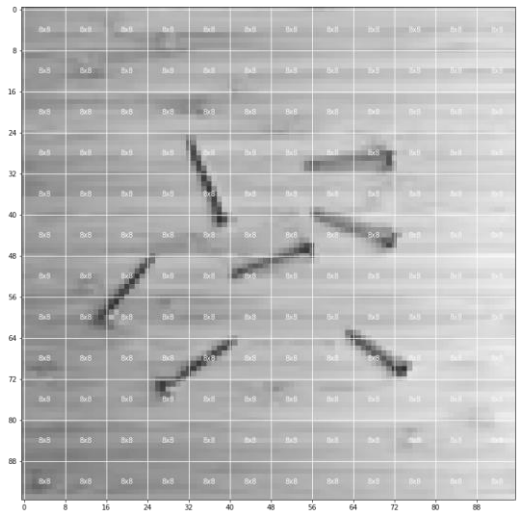
# 4. Faster Objects, More Objects (FOMO)




https://docs.edgeimpulse.com/docs/edge-impulse-studio/learning-blocks/object-detection/fomo-object-detection-for-constrained-devices

# 4. Training FOMO model

| layer | tensor shape | #parameters | comment |
|---|---|---|---|
| input | (None, 160, 160, 3) | 0 | input |
| MobileNet-V2 backbone | (None, 20, 20, 96) | 15,840 | image feature extractor |
| dense (Conv2D) | (None, 20, 20, 32) | 3,104 | fully convolutional classi |
| logits (Conv2D) | (None, 20, 20, 2) | 66 | fully convolutional classi |



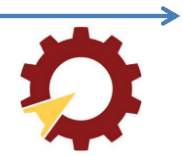
320 x 320 pixels → 12 x 12 heatmap

# 4. End-to-End edge vision workflow

1. Create project → target ESP32-EYE
2. Train model
   1. Grab image and label bounding box → EI Ingestion API
   2. Create and train model → FOMO model
   3. Deploy as C++ project → Arduino firmware
3. Create inference firmware
   1. Add C++ as library
   2. Grab image, convert to bitmap, resize to feature size
   3. Provide features via callback → `signal_t` struct
   4. Run classifier → `run_classifier(&signal, &result, false);`
   5. Interpret result → `result.`

https://www.survivingwithandroid.com/tinyml-esp32-cam-edge-image-classification-with-edge-impulse/

# Practice #4



1. Open project tgr2023_07_vision
2. Extract Edge Impulse firmware into /lib
3. Modify code to grab image, prepare feature, classify, and see result
4. Take snapshot and check result