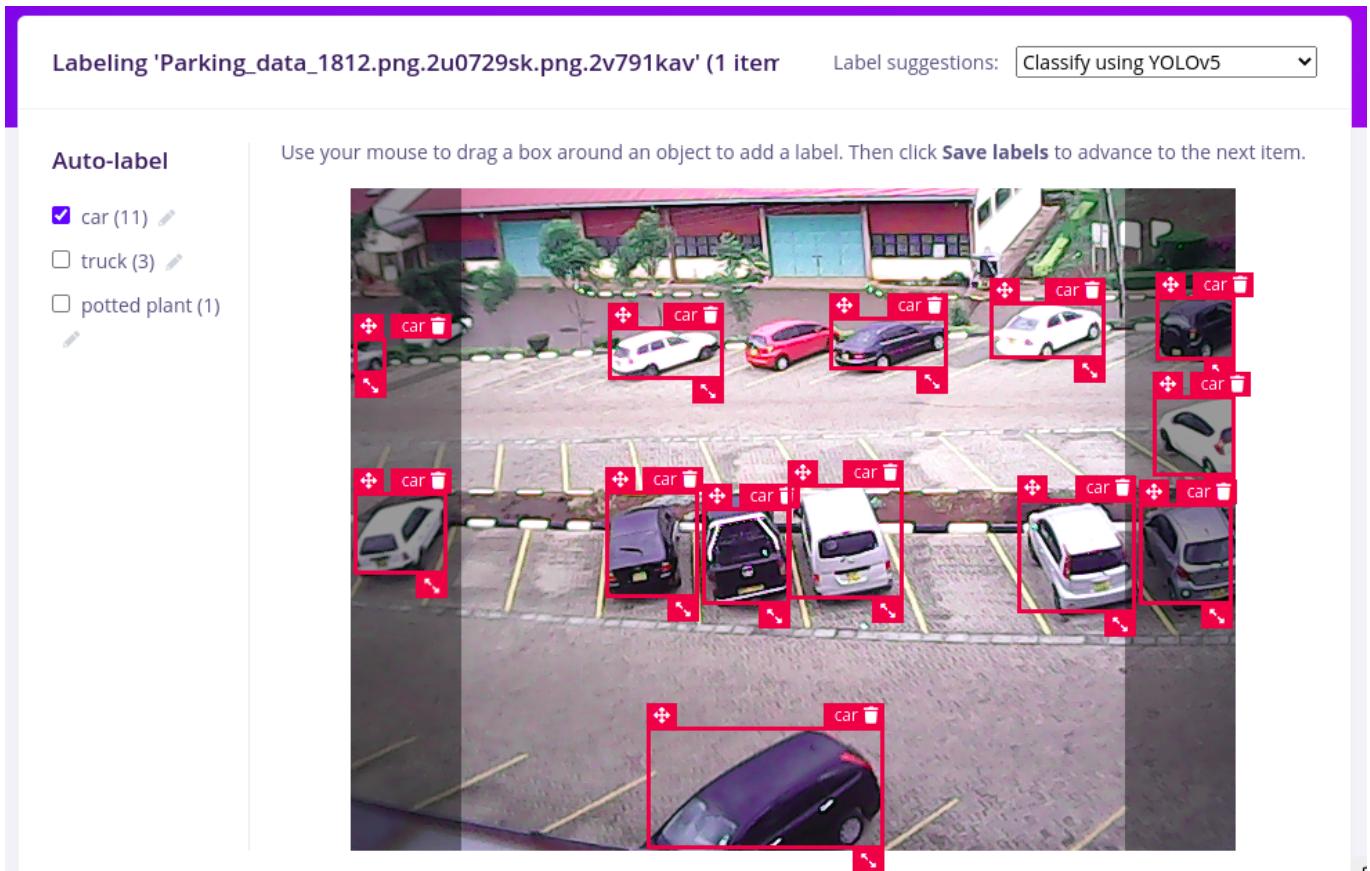


⌚ 엣지 디바이스에서 객체 인식하기 with Edge Impulse FOMO

이 문서에서는 Edge Impulse의 FOMO(Object Detection) 알고리즘을 이용해 마이크로컨트롤러 기반 엣지 디바이스에서 객체를 실시간으로 인식하는 전체 과정을 소개합니다.

⌚ FOMO란?

FOMO (Faster Objects, More Objects)는 Edge Impulse가 만든 객체 인식 알고리즘입니다.



▲ FOMO (Faster Objects, More Objects) - 엣지 디바이스용 실시간 객체 감지

- YOLO/SSD와 다르게 **Bounding Box** 대신 **중심점(Centroid)** 을 예측합니다.
- **200KB 이하 RAM**에서도 구동 가능 (ESP32, STM32 등)
- **YOLO보다 30배 빠름**, 소형 장치에서도 객체 추적·카운팅 지원

한계점

- 객체의 크기 정보는 없음
- 객체가 너무 불어있으면 정확도 저하 가능
- 유사한 크기의 객체에 최적화됨

📦 준비물

- Edge Impulse 계정

- ESP32-CAM, Raspberry Pi, Jetson Nano 등 지원되는 장치
- 또는 스마트폰/카메라 → 이미지 업로드

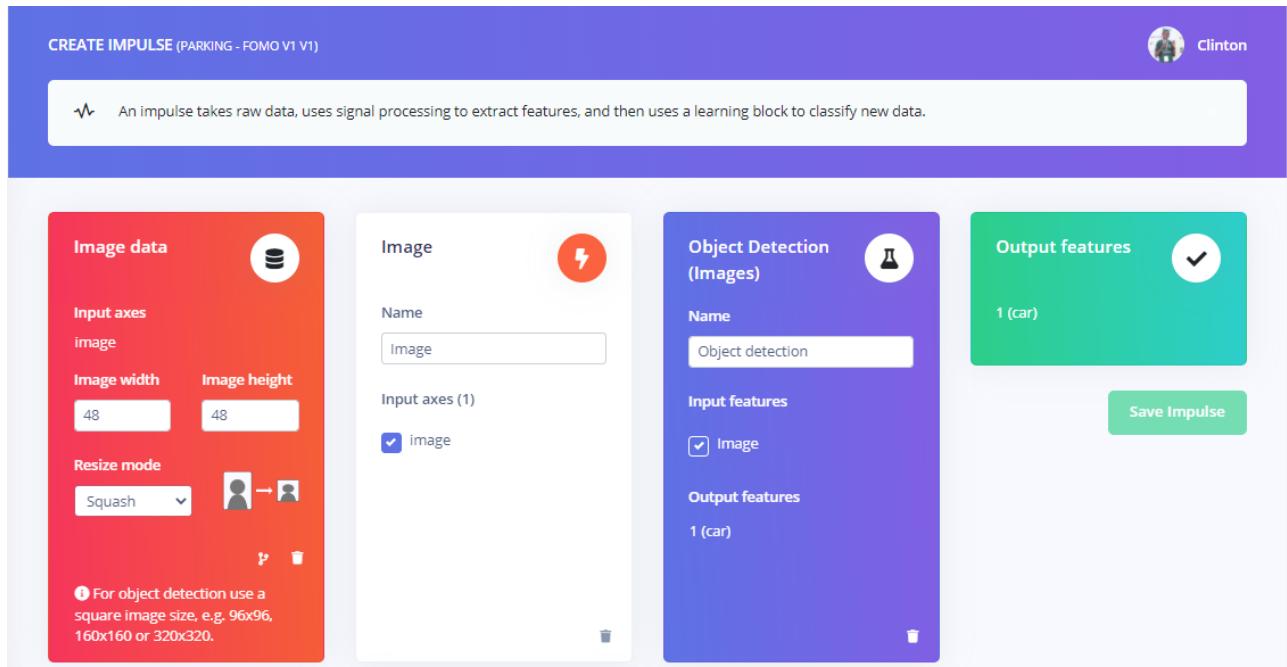
① 데이터 수집 및 라벨링

- Edge Impulse Studio에서 **Data acquisition** → **Upload** 또는 장치 연결로 데이터 수집
- **Labeling queue**에서 **Bounding box** 기반 라벨링
- YOLOv5, 기존 모델, Object tracking 기반 **AI Assisted Labeling**도 가능

예: 자동차 데이터 → YOLOv5로 90% 이상 자동 라벨링 가능

② Impulse 구성

1. **Create Impulse** 탭에서:
 - Input: Image (96×96, Grayscale)
 - Resize mode: Fit shortest axis
 - Add blocks: **Images** + **Object Detection (Images)**
2. **Save Impulse**



▲ 엣지 디바이스에서의 실시간 AI 추론

③ 전처리 및 특징 추출

- **Images** → RGB/Grayscale 선택 후 **Save Parameters**
- **Generate Features** 버튼 클릭
- Feature Explorer를 통해 **클러스터 구조** 시각화

EDGE IMPULSE

IMAGE (PARKING - FOMO V1 V1)
#1 Click to set a description for this version

Parameters Generate features

Raw data

Parking_data_1812.png 2u0729sk (car, ▾)

Raw features ⓘ

0xb07894, 0xa896a7, 0xeeceff6, 0xe4cd6, 0xcc8ca4, 0xd791ab, 0xd791ac, 0xdct9b1, 0xclab9, 0xcbab5f, 0xcbacc4, 0xcb8ac...

Parameters

Image

Color depth Grayscale

Save parameters

DSP result

Image

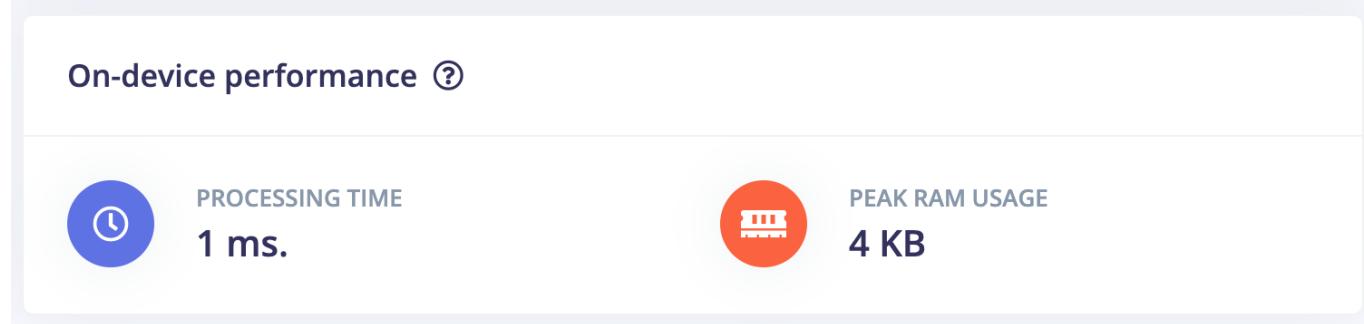
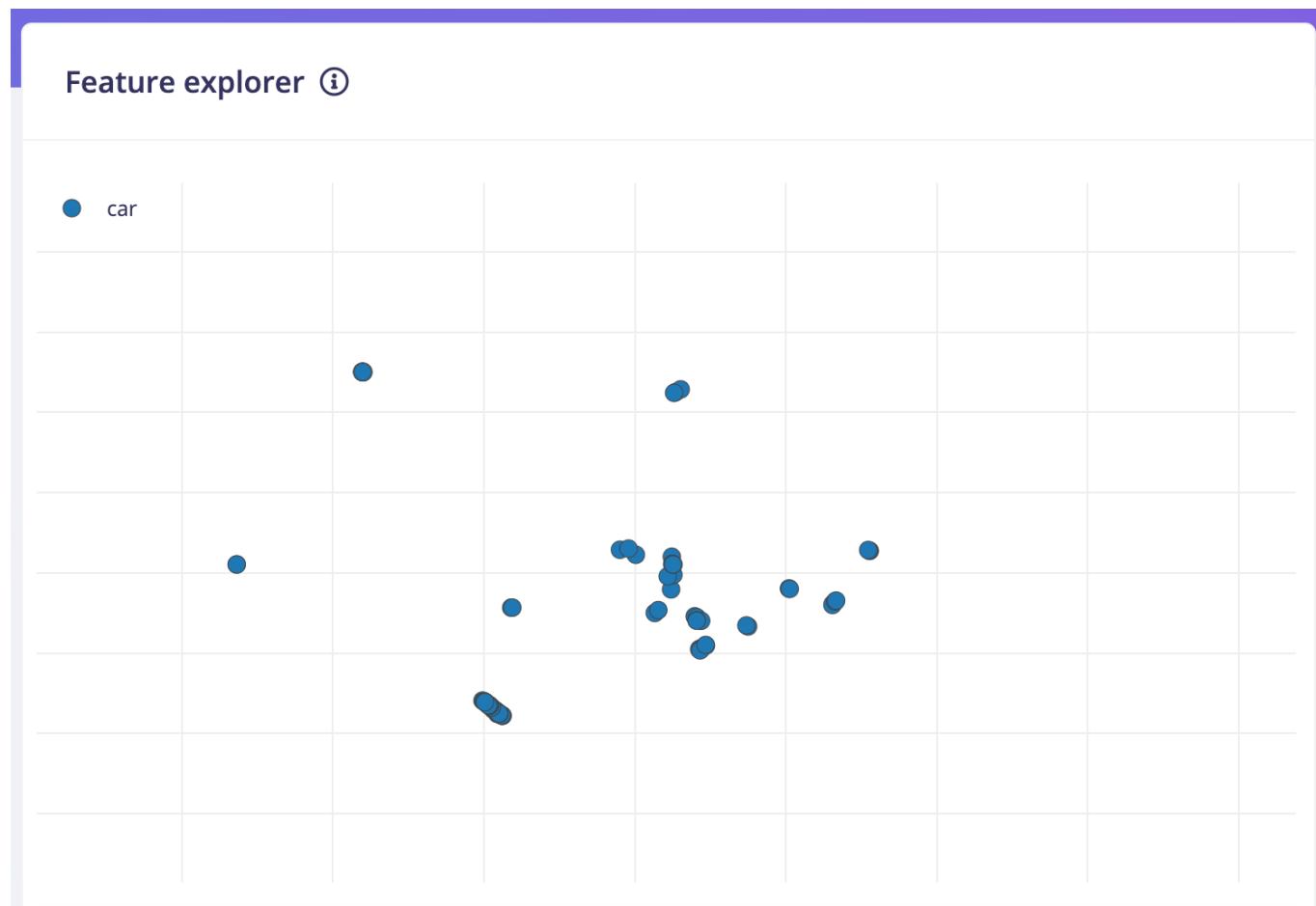
Processed features ⓘ

0.5488, 0.6169, 0.9392, 0.8372, 0.6348, 0.6623, 0.6674, 0.6893, 0.7172, 0.7441, 0.7591, 0.7585, 0.7516, 0.748...

On-device performance ⓘ

PROCESSING TIME 1 ms.

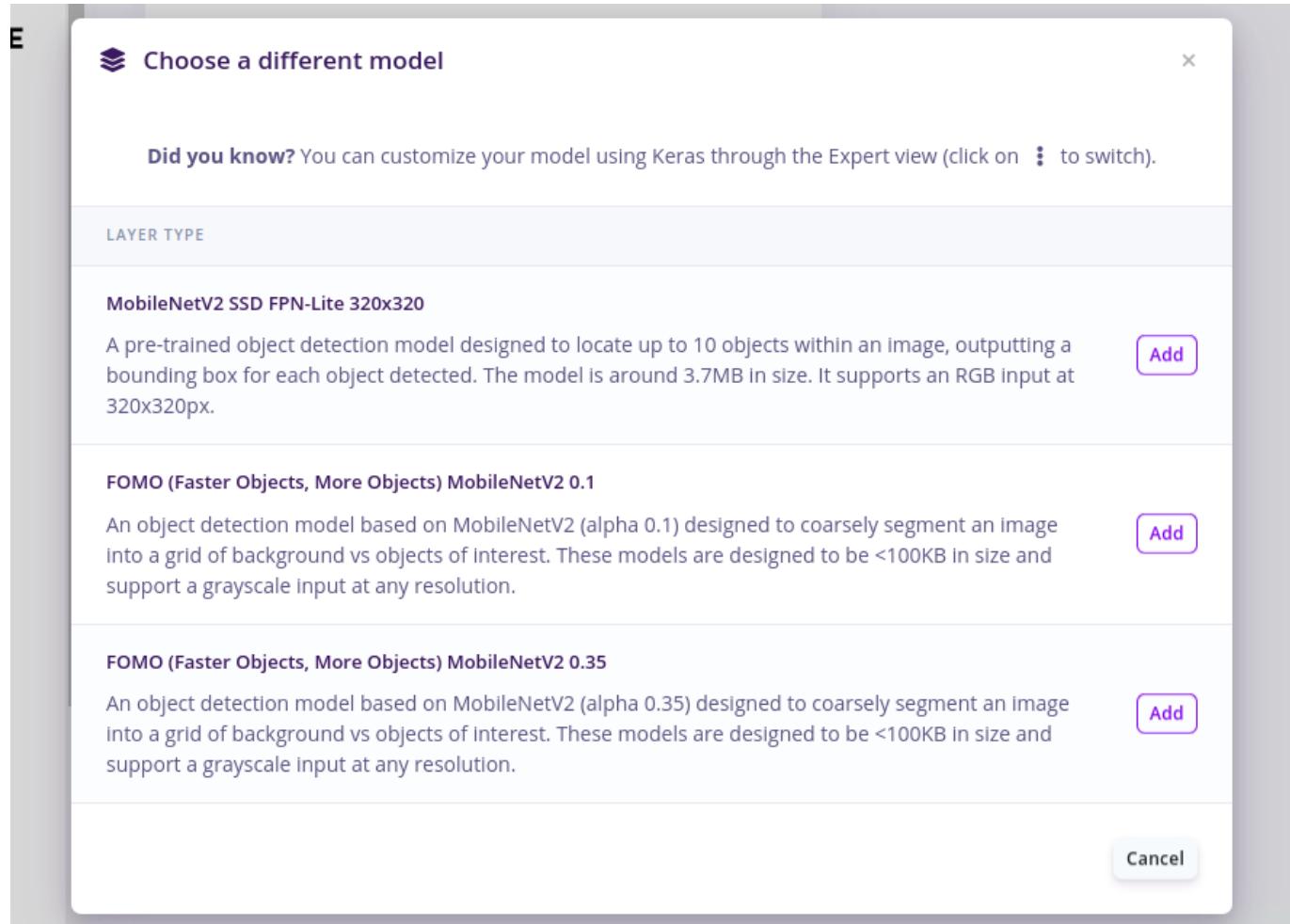
PEAK RAM USAGE 4 KB



▲ Edge Impulse Studio의 데이터 라벨링 시스템

④ 모델 학습

- Object Detection 탭 → Choose a different model → FOMO 선택
- Learning Rate: 0.001 권장
- 학습 완료 후 **F1 Score, Precision, Recall** 등 확인 가능
- FOMO는 모바일넷 V2 기반으로 전이학습 가능



Model Model version: ② Quantized (int8) ▾

Last training performance (validation set)

 F1 SCORE
85.5%

Confusion matrix (validation set)

	BACKGROUND	CAR
BACKGROUND	99.5%	0.5%
CAR	22.9%	77.1%
F1 SCORE	0.98	0.85

On-device performance ②

 INFEREN... -  PEAK RA... 243.9K  FLASH U... 77.5K

⑤ 테스트 및 Live Classification

- 수집한 데이터 중 20%는 테스트셋으로 분리
- Model Testing → Classify All
- Live Classification → 예측 결과 이미지와 오버레이 비교

예시:

- 실제 차량 18대 중 16대 정확히 탐지 → Precision 100%, Recall 63.6%, F1 Score 77.78%

Test data

Set the 'expected outcome' for each sample to the desired outcome to automatically score the impulse.

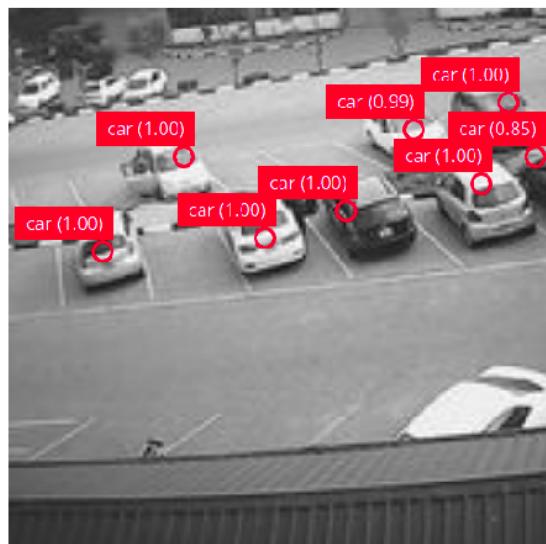
SAMPLE...	EXPECTED ...	LE...	F1 SCO...	RESULT
Parkin...	car, car, c...	-	74%	...
Parkin...	car, car, c...	-	94%	...
Parkin...	car, car, c...	-	85%	...
Parkin...	car, car, c...	-	90%	...
Parkin...	car, car, c...	-	83%	...
Parkin...	car, car, c...	-	87%	...

Model testing output

Model testing results

%
ACCURACY ⓘ
84.62%

Demo Team / Parking - FOMO



▲ FOMO를 활용한 주차장 점유율 실시간 모니터링

⑥ 디바이스 배포 및 실시간 실행

- Deployment 탭 → Arduino Library 또는 Linux Binary 생성
- Arduino IDE:
 - Sketch → Include Library → Add .ZIP
 - 예제 실행: Examples from Custom Library → 업로드
- Raspberry Pi:

edge-impulse-linux-runner

웹에서 실시간 확인:

`http://[장치IP]:[포트]`

예) <http://192.168.8.117:4912>

예제 코드 (Arduino with ESP32-CAM)

```
#include <whiteball_inferencing.h>
...
bool ei_camera_capture(...) {
    ...
    run_classifier(&signal, &result, false);
    for (int i = 0; i < result.bounding_boxes_count; i++) {
        ei_printf("%s (%f) [x:%u, y:%u, w:%u, h:%u]\n",
                  result.bounding_boxes[i].label,
                  result.bounding_boxes[i].value,
                  result.bounding_boxes[i].x,
                  result.bounding_boxes[i].y,
                  result.bounding_boxes[i].width,
                  result.bounding_boxes[i].height);
    }
}
```

The terminal window displays the following text:

```

| Starting inferencing in 2 seconds...
| taking photo...
'redetections (DSP: 1 ms., Classification: 72 ms., Anomaly: 0 ms.):
car (0.972656) [ x: 32, y: 24, width: 8, height: 24 ]
car (0.984375) [ x: 16, y: 40, width: 8, height: 8 ]
car (0.679687) [ x: 24, y: 56, width: 8, height: 8 ]
car (0.996094) [ x: 56, y: 56, width: 16, height: 16 ]
car (0.933594) [ x: 88, y: 64, width: 8, height: 8 ]
car (0.996094) [ x: 24, y: 72, width: 16, height: 24 ]
car (0.996094) [ x: 8, y: 80, width: 8, height: 8 ]
car (0.992187) [ x: 56, y: 80, width: 16, height: 16 ]

Starting inferencing in 2 seconds...

```

The code editor shows the following C++ code:

```

 Autoscroll  Show timestamp





```

ei_printf("Camera initialized\r\n");
}

for (size_t ix = 0; ix < ei_dsp_blocks_size; ix++) {
 ei_model_dsp_t block = ei_dsp_blocks[ix];
 if (block.extract_fn == &extract_image_features) {
 ei_dsp_config_image_t config = *((ei_dsp_config_image_t*)block.config);
 int16_t channel_count = strcmp(config.channels, "Grayscale") == 0 ? 1 : 3;

```


```

Terminal output (highlighted in green):

```

downloading element to address = 0x00040000, size = 234936
Erase      [=====] 100%      234936 bytes
Erase done.
Download   [=====] 100%      234936 bytes
Download done.
File downloaded successfully
Transitioning to dfuMANIFEST state

```

요약

항목	설명
목적	초저가 엣지 디바이스에서 실시간 객체 인식
알고리즘	FOMO (Centroid 기반, 경량)
장점	빠르고 가벼움, 실시간 처리 가능
한계	Bounding box 미포함, 객체 크기 알 수 없음
실습 흐름	데이터 수집 → Impulse 설계 → 학습 → 테스트 → 디바이스 배포

참고 링크

- FOMO 공식 문서
- Edge Impulse Studio
- Arduino 라이브러리 배포 안내