Online Object Detection

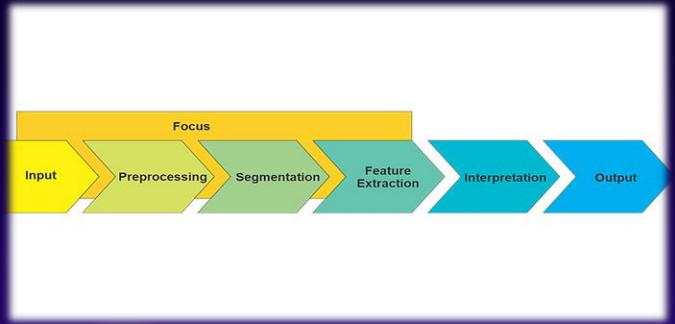
(Image Processing)

M.M.Mirzaei

Instructor : Ms.Kamali.Yazdi

Its all about digital image processing (DIP)

Digital Image Processing means processing digital image by means of a digital computer. We can also say that it is a use of computer algorithms, in order to get enhanced image either to extract some useful information.



Requires the knowledge of basic calculus, probability, and differential equations for mathematical calculation purposes. Also, there is a basic knowledge of programming like MATLAB, C++, etc.

What is an Digital Image?

Projection of 3D scene in 2D plane.

An image is defined as a two-dimensional function,**F(x,y)**, where x and y are spatial coordinates, and the amplitude of **F** at any pair of coordinates (x,y) is called the **intensity** of that image at that point. When x,y, and amplitude values of **F** are finite, we call it a **digital image**.

1. **BINARY IMAGE**– The binary image as its name suggests, contain only two pixel elements i.e 0 & 1,where 0 refers to black and 1 refers to white. This image is also known as Monochrome.

2. **BLACK AND WHITE IMAGE**– The image which consist of only black and white color is called BLACK AND WHITE IMAGE.

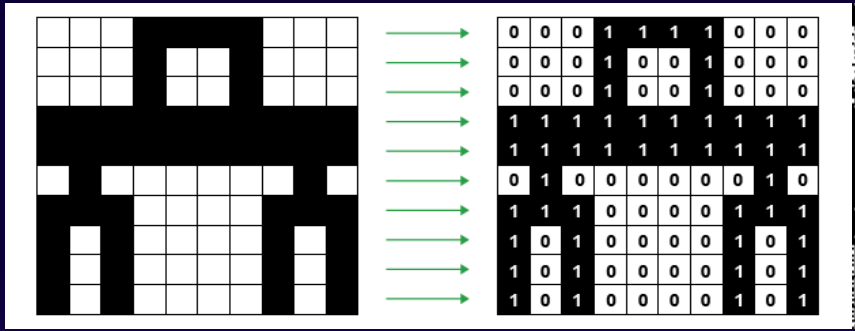
3. **8 bit COLOR FORMAT**– It is the most famous image format.It has 256 different shades of colors in it and commonly known as Grayscale Image. In this format, 0 stands for Black, and 255 stands for white, and 127 stands for gray.

4. **16 bit COLOR FORMAT**– It is a color image format. It has 65,536 different colors in it. It is also known as High Color Format. In this format the distribution of color is not as same as Grayscale image.

- 4.1 RGB - A pixel is made up of 3 integers between 0 to 255 (the integers represent the intensity of red, green, and blue).

- 4.2 RGBA - It is an extension of RGB with an added alpha field, which represents the opacity of the image.

BINARY IMAGE



Black and White Image

8 bit Color Format
16 bit Color Format

Difference between
8-bit and 16 bit color format in DIP

8-bit Color Representation

16-bit Color Representation

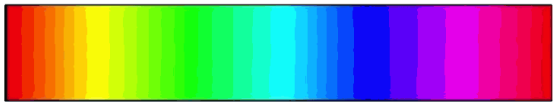
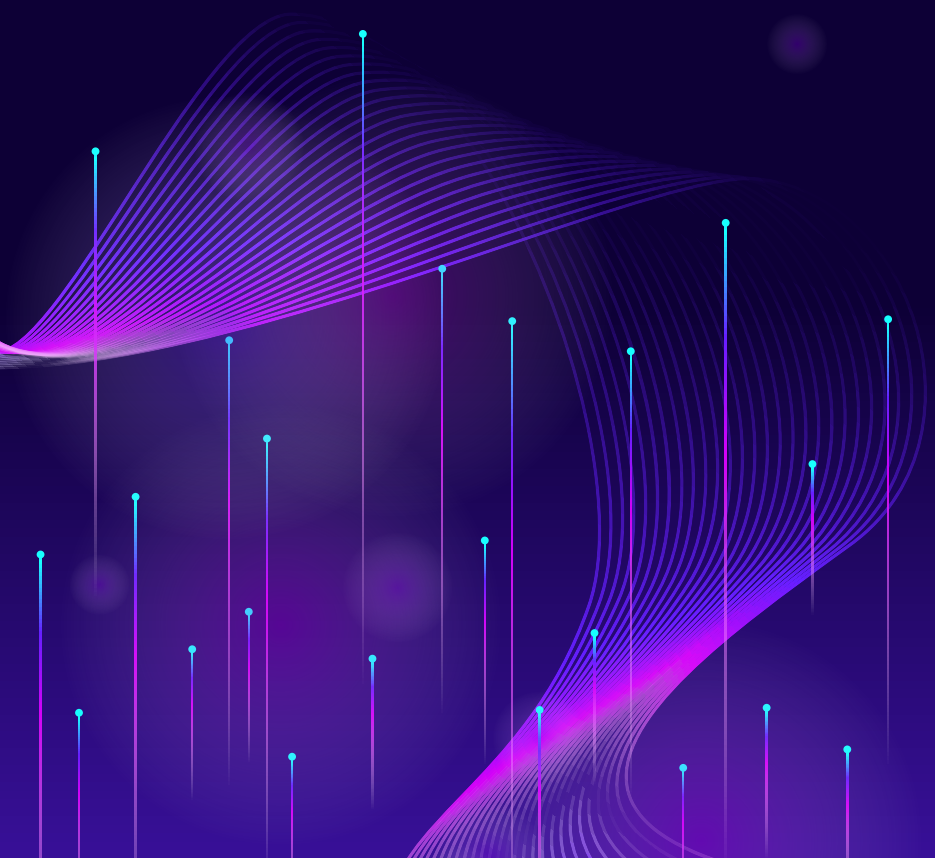


Image Processing Steps



- Image Acquisition
- Image Enhancement
- Image Restoration
- Color Image Processing
- Image Segmentation

01

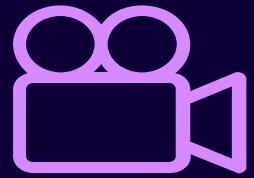
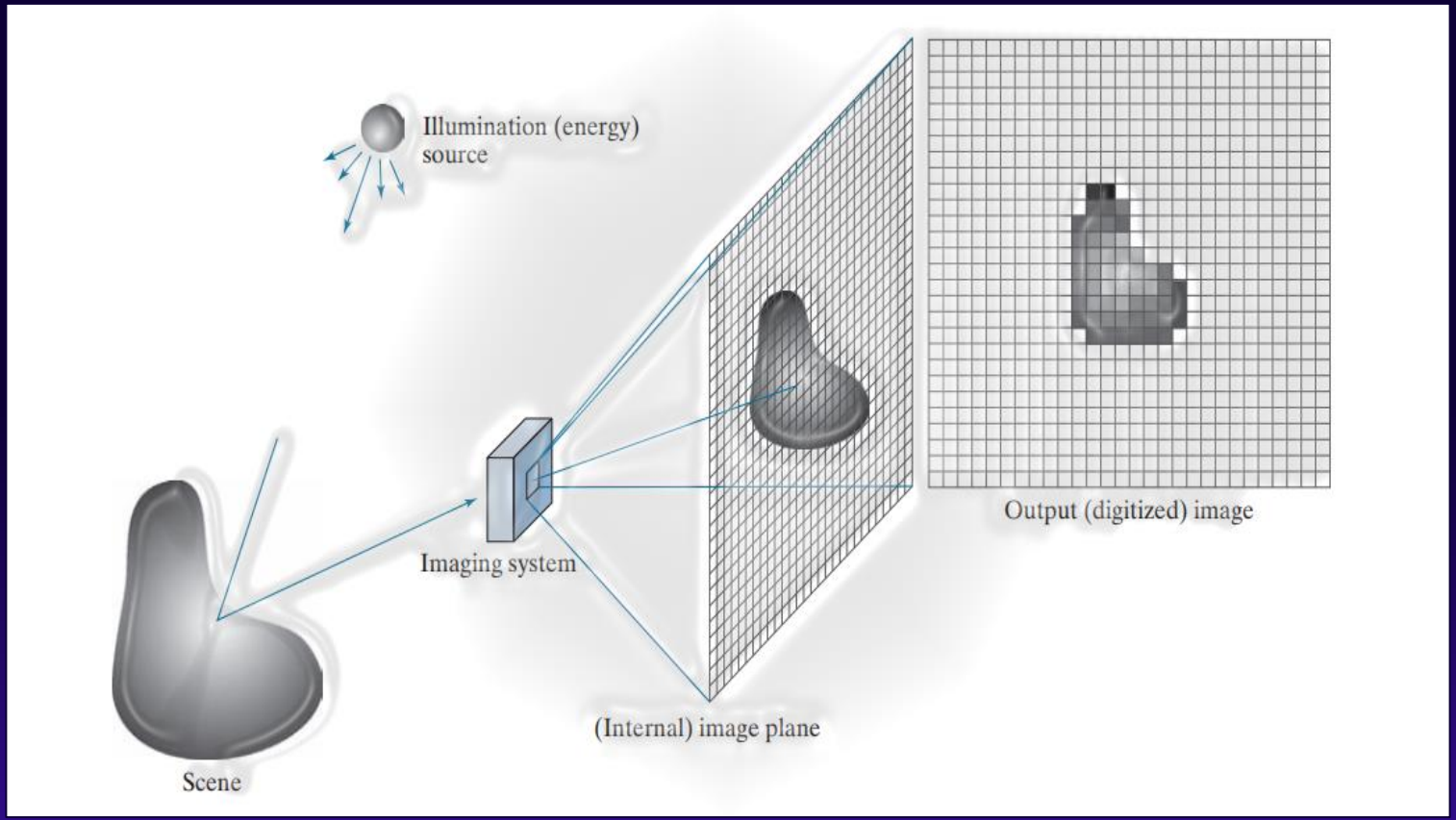
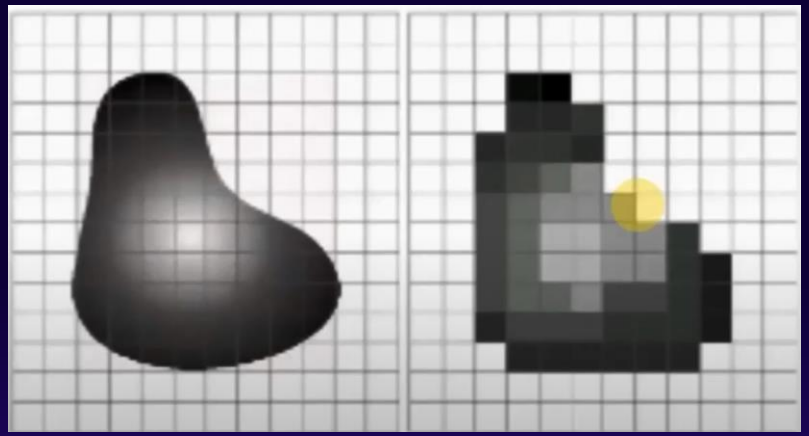


Image Acquisition

The process of capturing visual information from the real world and converting it into a digital image that can be processed by computers


Illumination (energy)
source
Imaging system
Scene
(Internal) image plane
Output (digitized) image

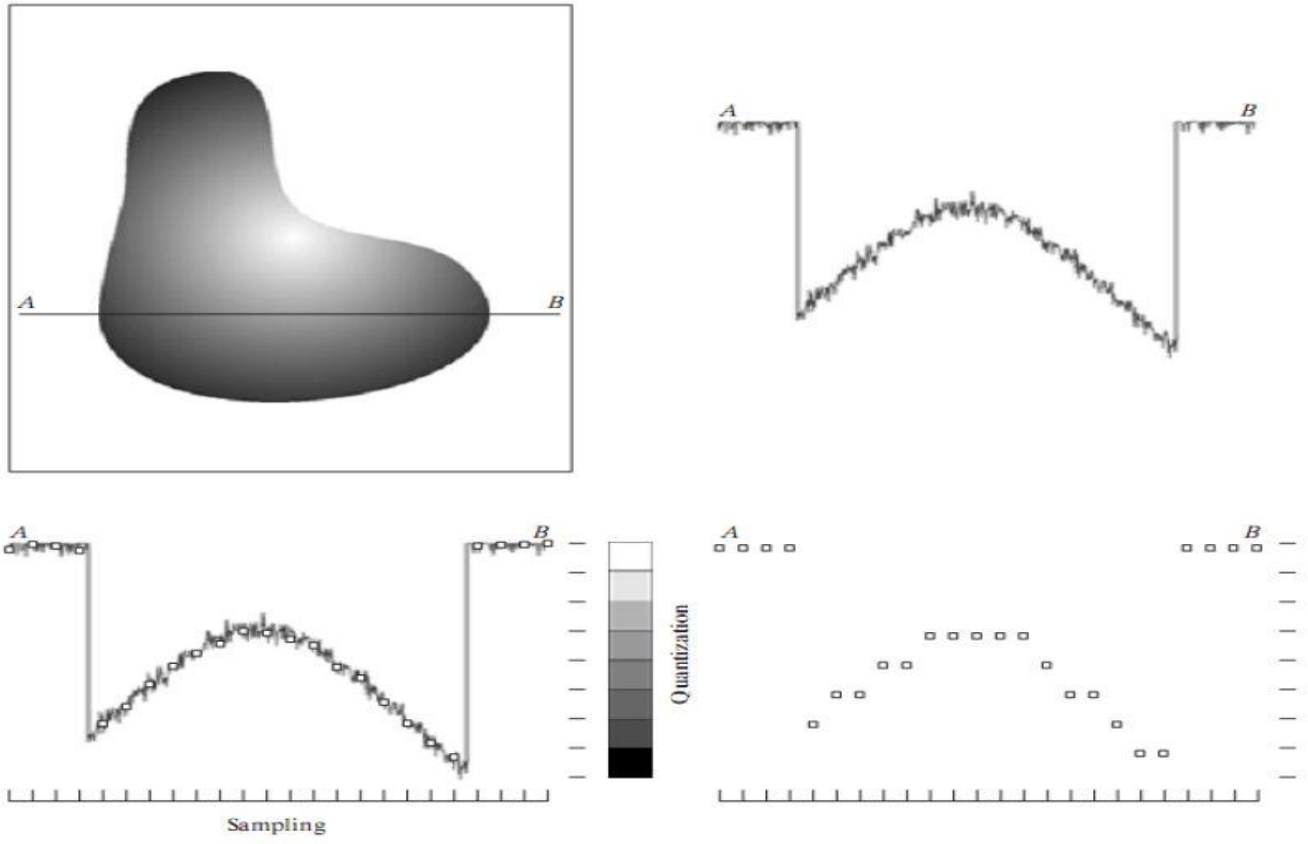
Convert Analoge Image To Digital Image



Digitizing an Image

Sampling

Quantization


A
B
A
B
A
B
Sampling
Quantization
A
B

02

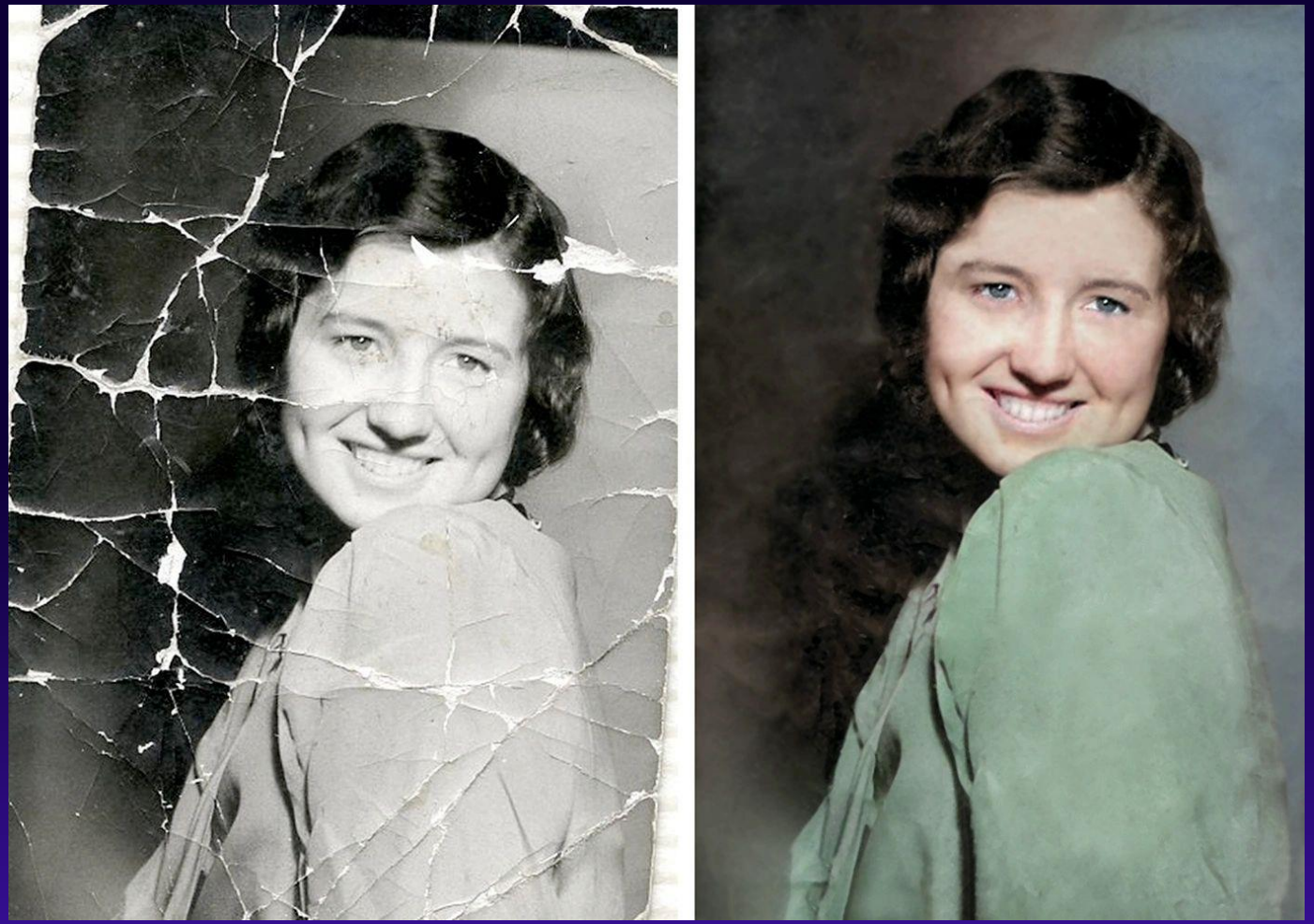
Image Enhancement

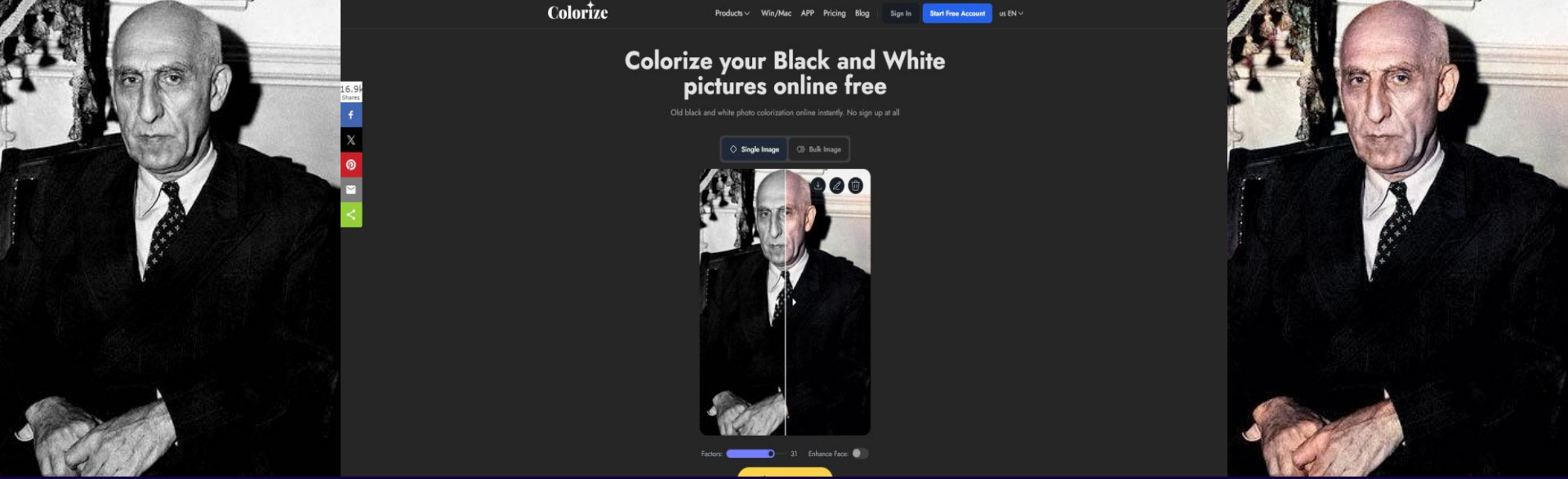
In this step, the acquired image is manipulated to meet the requirements of the specific task for which the image will be used. Such techniques are primarily aimed at highlighting the hidden or important details in an image, like contrast and brightness adjustment, etc. Image enhancement is highly subjective in nature..

03

Image Restoration

Image restoration is the process of improving the appearance of an image. However, unlike image enhancement, image restoration is done using certain **mathematical** or **probabilistic** models.





https://imagecolorizer.com/colorize

04

Color Image Processing

This step aims at handling the processing of colored images (16-bit RGB or RGBA images), for example, peforming color correction or color modeling in images.

=

Blue

Green

Red

123	94	83	2
123	94	83	4
123	94	83	2
34	44	187	92
34	76	232	124
67	83	194	202

30							
92							
124							
4							
142							
4							

pad

pad

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	123	94	83	2	0	0
0	0	34	44	187	92	0	0
0	0	34	76	232	124	0	0
0	0	67	83	194	202	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

pad

pad

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	123	94	2	4	0	0
0	0	11	3	22	192	0	0
0	0	12	4	23	34	0	0
0	0	194	83	12	94	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

pad

pad

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	123	94	83	2	0	0
0	0	34	44	37	30	0	0
0	0	34	114	234	124	0	0
0	0	49	18	204	142	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

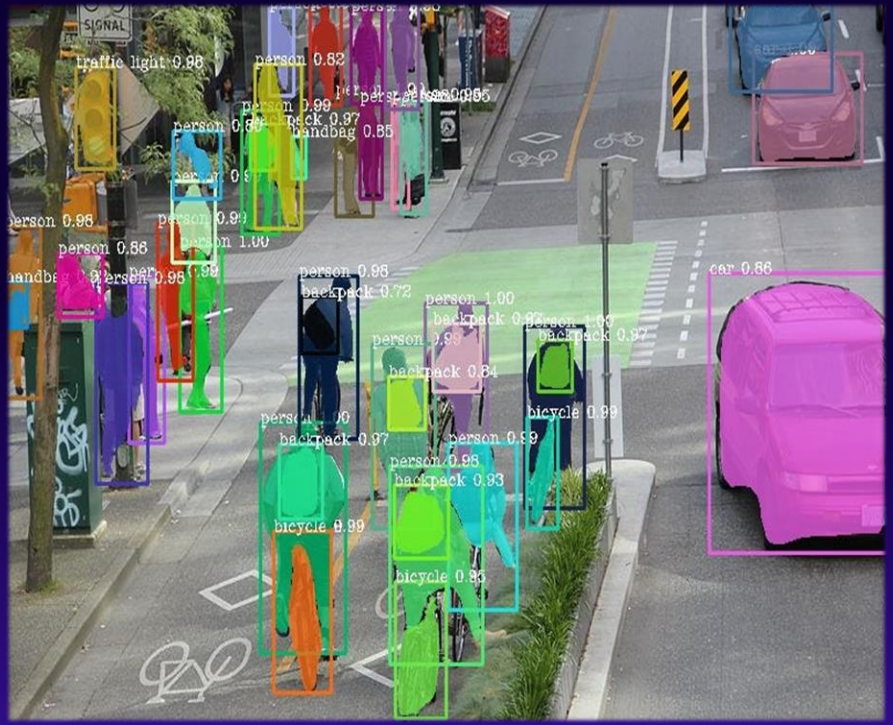
05

Image Segmentation

Divides an image into distinct groups of pixels, called segments, to help with object detection and other tasks. It's a key step in image analysis and processing, and is used in many fields, including medical imaging, autonomous driving, and background removal

Image Segmentation

- **Thresholding**
- Clustering methods
- Dual clustering method
- Motion and interactive segmentation
- Compression-based methods
- Histogram-based methods
- **Edge detection**
- Isolated Point Detection
- Region-growing methods
- Variational methods



Object Recognition Methods

Object Recognition Using Machine Learning

•**HOG (Histogram of oriented Gradients) feature Extractor and SVM (Support Vector Machine) model**
Before the era of deep learning, it was a state-of-the-art method for object detection. It takes histogram descriptors of both **positive (images that contain objects)** and negative (images that does not contain objects)

•**Viola-Jones algorithm:** This algorithm is widely used for face detection in the image or real-time. It performs Haar-like feature extraction from the image. This generates a large number of features. These features are then passed into a boosting classifier.

Viola-Jones is that it has a detection time of 2 fps which can be used in a real-time face recognition system.

Object Recognition Using Deep Learning

Convolution Neural Network (CNN) is one of the most popular ways of doing object recognition. It is widely used and most state-of-the-art neural networks used this method for various object recognition related tasks such as image classification. This CNN network takes an image as input and outputs the probability of the different classes. If the object present in the image then it's output probability is high else the output probability of the rest of classes is either negligible or low. The advantage of Deep learning is that we don't need to do feature extraction from data as compared to machine learning.

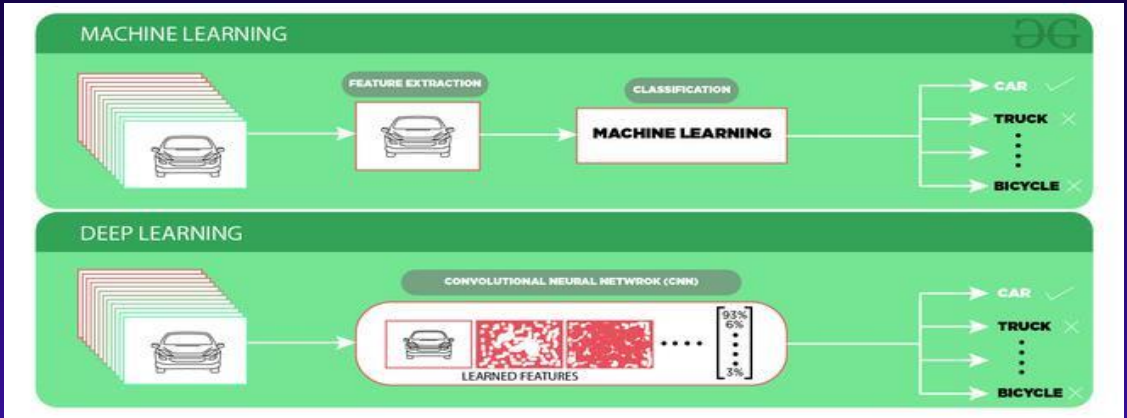
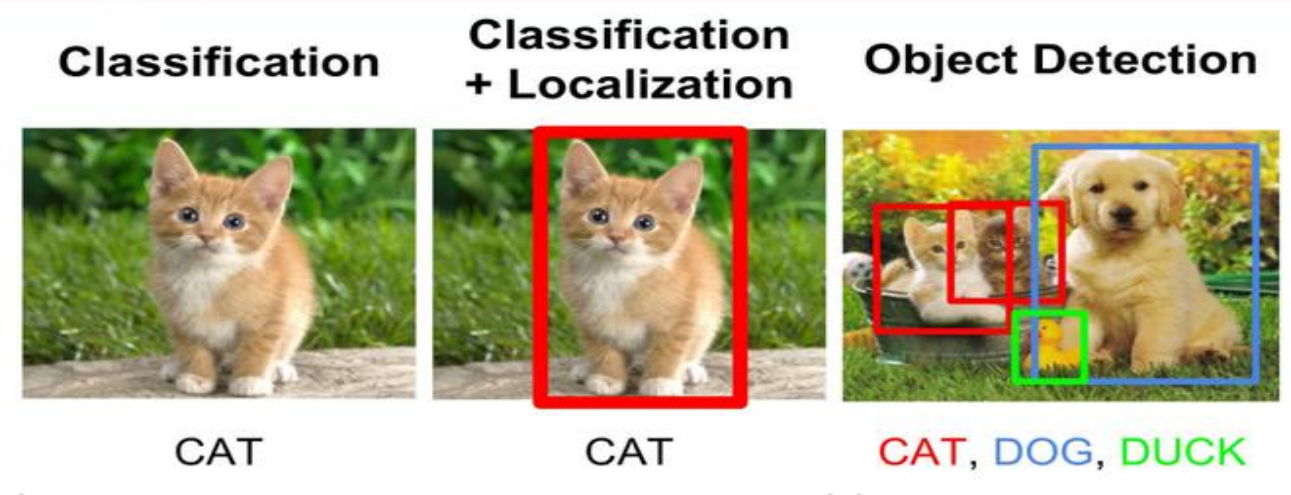


Image Classification

In Image classification, it takes an image as an input and outputs the classification label of that image with some metric (probability, loss, accuracy, etc). For Example: An image of a cat can be classified as a class label “cat” or an image of Dog can be classified as a class label “dog” with some probability.

Object Localization

This algorithm locates the presence of an object in the image and represents it with a bounding box. It takes an image as input and outputs the location of the bounding box in the form of (position, height, and width).



What is an object detection?

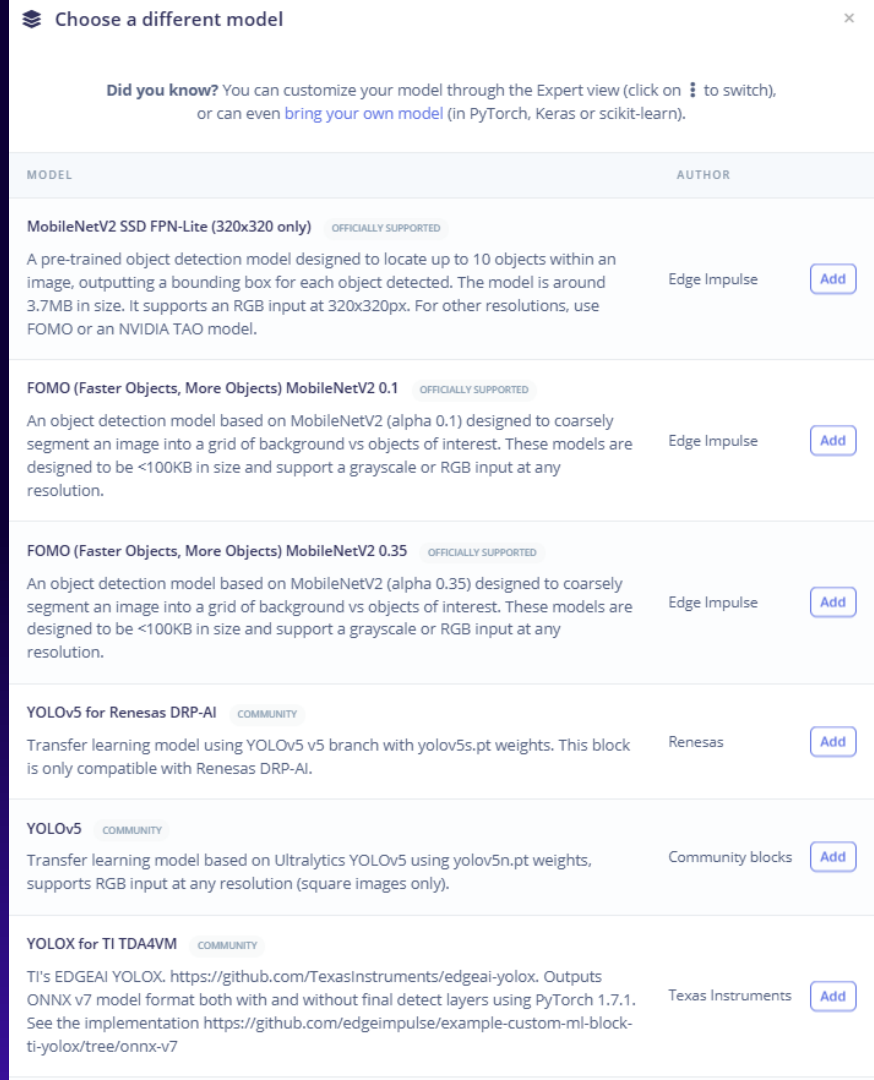
Object detection is the process of taking a machine that has a camera on it and training it on one or more different objects afterwards that machine will be able to recognize the objects and report back the position of the object or objects within its field of view.

Object Detection algorithms act as a **combination of image classification and object localization.** It takes an image as input and produces one or more bounding boxes with the class label attached to each bounding box. These algorithms are capable enough to deal with multi-class classification and localization as well as to deal with the objects with multiple occurrences.

Object Detection But real time on Microcontroller

It was only a few years ago that you needed **huge computers** in order to perform this task and now we can actually do it on **microcontrollers**

Different Model



Mobilenet Model

Mobilenet is a type of **convolutional neural network** designed for mobile and embedded vision applications. Instead of using standard convolution layers, they are based on a streamlined architecture that uses depthwise separable convolutions.

Using this architecture, we can build lightweight deep neural networks that have low latency for **mobile and embedded devices (example: jetson nano).**

YOLO Model

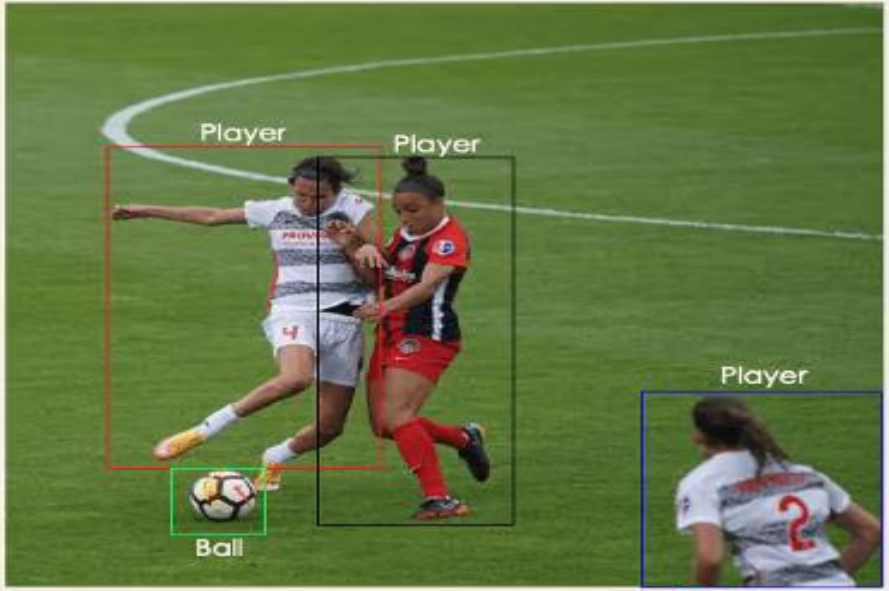
YOLO (You Only Look Once) is a **real-time object detection** algorithm developed by **Joseph Redmon and Ali Farhadi** in 2015. It is a single-stage object detector that uses a **convolutional neural network (CNN)** to predict the bounding boxes and class probabilities of objects in input images.

The YOLO algorithm divides the input image into a grid of cells, and for each cell, it predicts the probability of the presence of an object and the bounding box coordinates of the object. It also predicts the class of the object. Unlike two-stage object detectors such as R-CNN and its variants, YOLO processes the entire image in one pass, making it faster and more efficient.

YOLO has been developed in several versions, such as YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, and YOLOv7. Each version has been built on top of the previous version with enhanced features such as improved accuracy, faster processing, and better handling of small objects.

(A) Input image

(B) YOLO Algorithm result



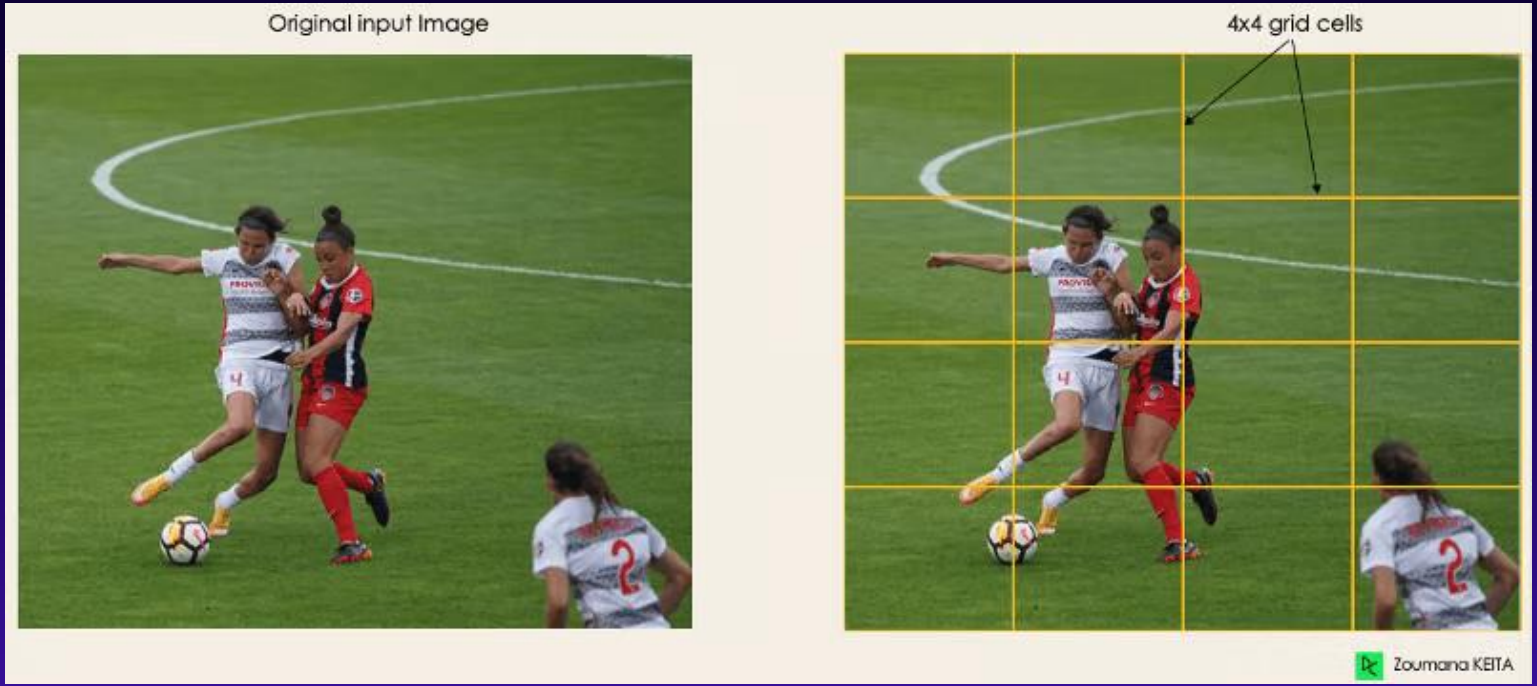
Zoumana KEITA

The algorithm works based on the following three approaches:

- Residual blocks
- Bounding box regression
- Intersection Over Unions or IOU for short

Residual blocks

This first step starts by dividing the original image (A) into NxN grid cells of equal shape, where N, in our case, is 4, as shown in the image on the right. Each cell in the grid is responsible for localizing and predicting the class of the object that it covers, along with the probability/confidence value.



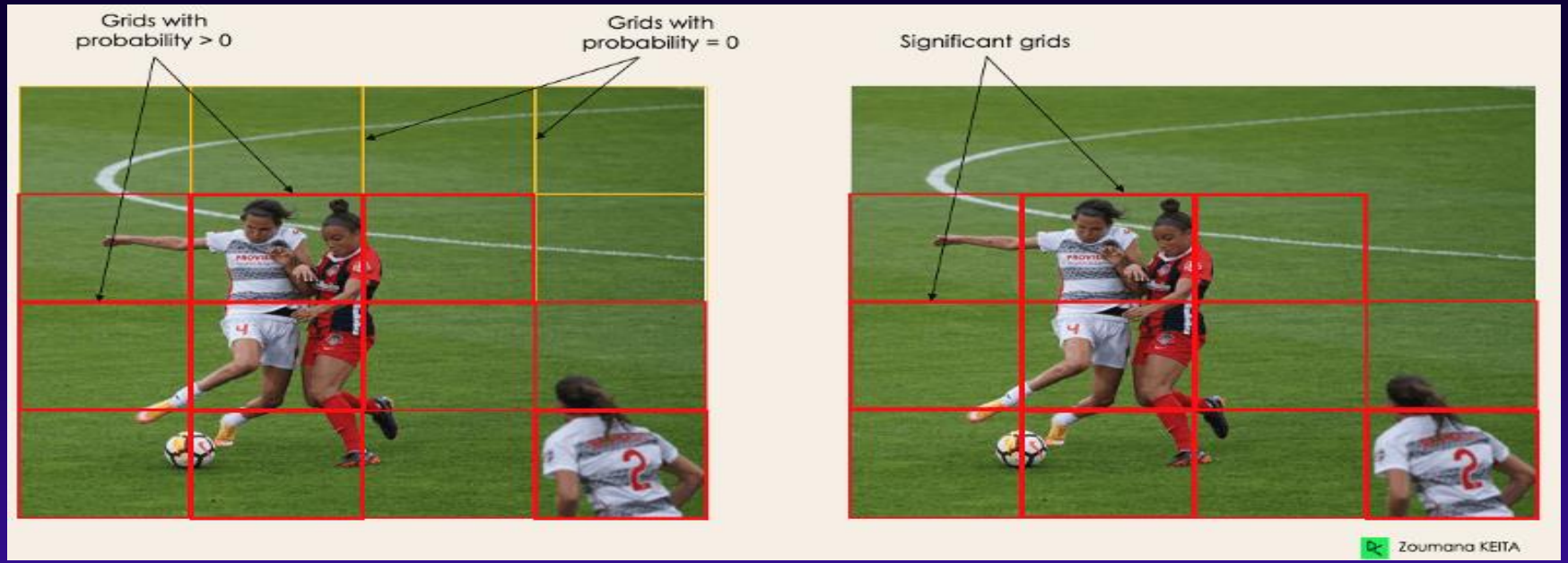
Bounding box regression

The next step is to determine the bounding boxes corresponding to rectangles, highlighting all the objects in the image. We can have as many bounding boxes as there are objects within a given image.

YOLO determines the attributes of these bounding boxes using a single regression module in the following format, where Y is the final vector representation for each bounding box.

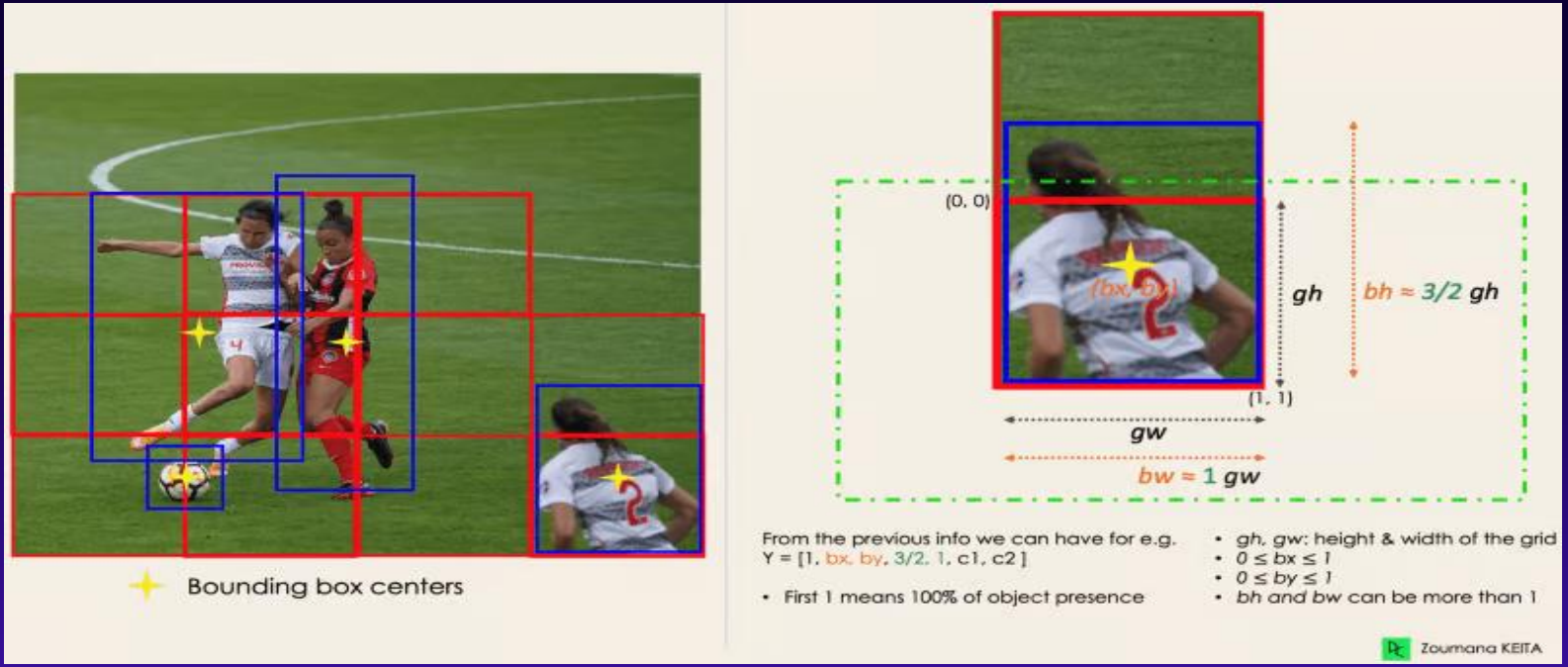
Y = [pc, bx, by, bh, bw, c1, c2]

- pc corresponds to the probability score of the grid containing an object. For instance, all the grids in red will have a probability score higher than zero. The image on the right is the simplified version since the probability of each yellow cell is zero (insignificant).



- bx, by are the x and y coordinates of the center of the bounding box with respect to the enveloping grid cell.
- bh, bw correspond to the height and the width of the bounding box with respect to the enveloping grid cell.

c1 and c2 correspond to the two classes, Player and Ball. We can have as many classes as your use case requires.



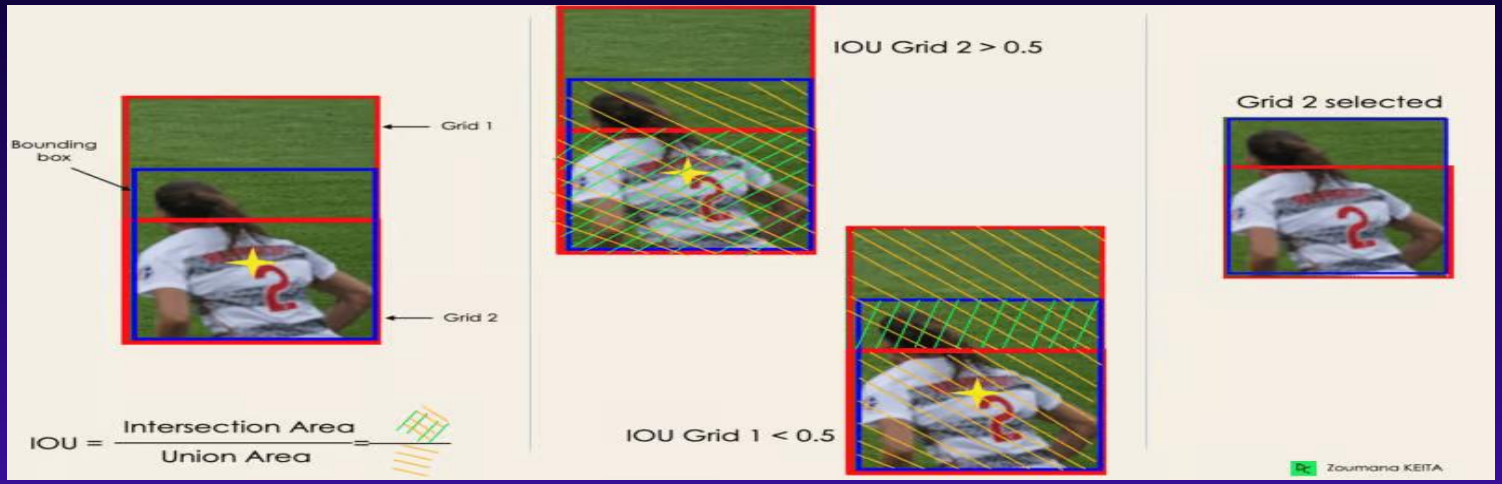
Intersection Over Unions or IOU

Most of the time, a single object in an image can have multiple grid box candidates for prediction, even though not all are relevant. The goal of the IOU (a value between 0 and 1) is to discard such grid boxes to only keep those that are relevant. Here is the logic behind it:

The user defines its IOU selection threshold, which can be, for instance, 0.5.

Then, YOLO computes the IOU of each grid cell, which is the Intersection area divided by the Union Area.

Finally, it ignores the prediction of the grid cells having an IOU ≤ threshold and considers those with an IOU > threshold.



Aplications

Healthcare

Agriculture

Security Surveillance

Self-driving cars

FOMO Model

This algorithm locates the presence of an object in the image and represents it with a bounding box. It takes an image as input and outputs the location of the bounding box in the form of (position, height, and width).

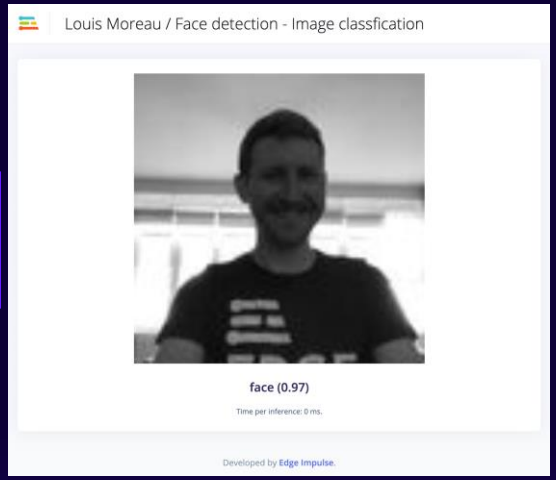
FOMO is a ground-breaking algorithm that brings real-time object detection, tracking and counting to microcontrollers for the first time. FOMO is 30x faster than MobileNet SSD and runs in <200K of RAM.

To give you an idea, we have seen results around 30 fps on the Arduino Nicla Vision (Cortex-M7 MCU) using 245K RAM.

The two most common image processing problems are image classification and object detection.

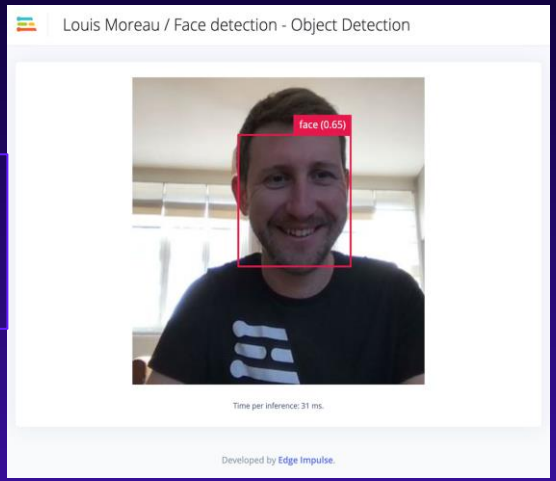
Image classification takes an image as an input and outputs what type of object is in the image. This technique works great, even on microcontrollers, as long as we only need to detect a single object in the image.

"Is there a face or not in the image?"



Object detection takes an image and outputs information about the class and number of objects, position, and size in the image.

"Are there faces in the image, where and what size are they?"

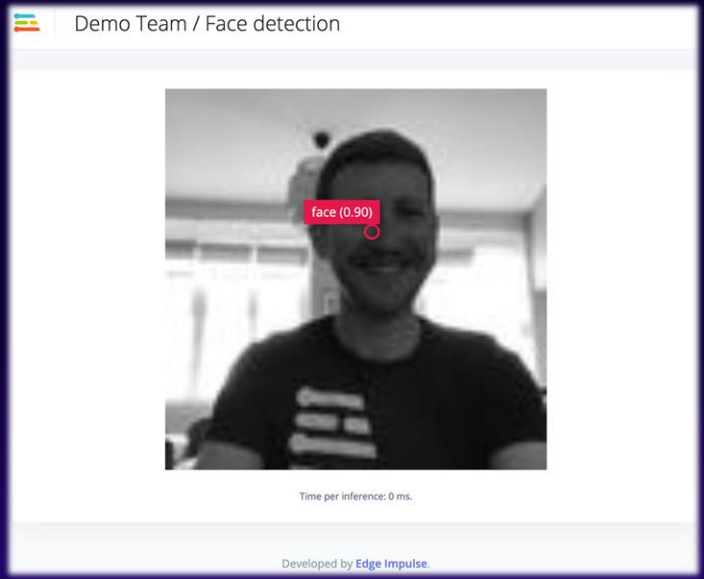


Trained on centroid

The main design decisions for **FOMO** are based on the idea that a lot of object detection problems don’t actually need the size of the objects but rather just their location in the frame.

Once we know the location of salient objects we can ask further questions such as "Is an object above/below another?" or "How many of these objects are in view?"

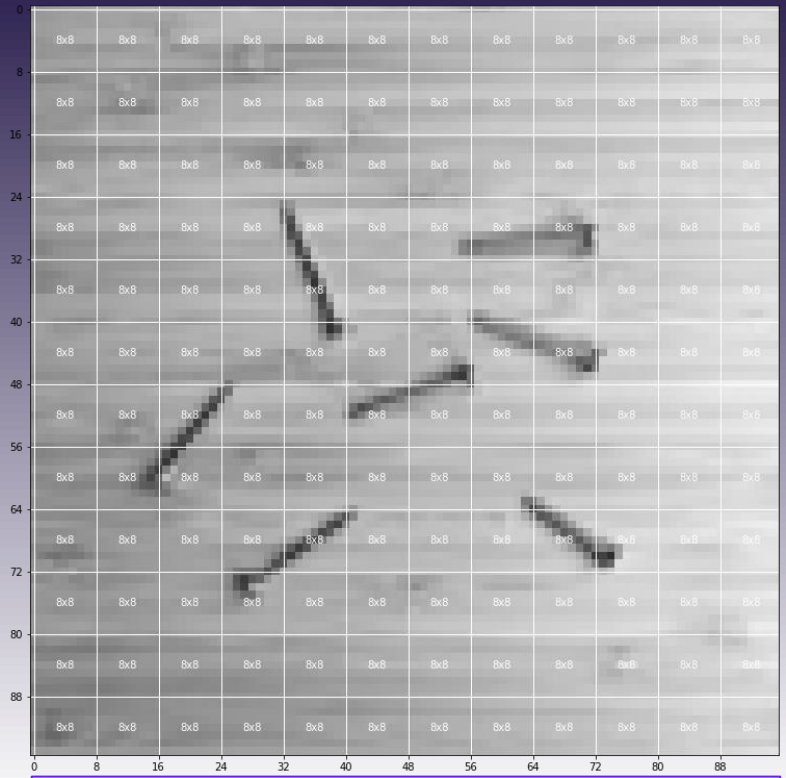
Based on this observation, classical bounding boxes are no longer needed. Instead, a detection based on the centroids of objects is enough.



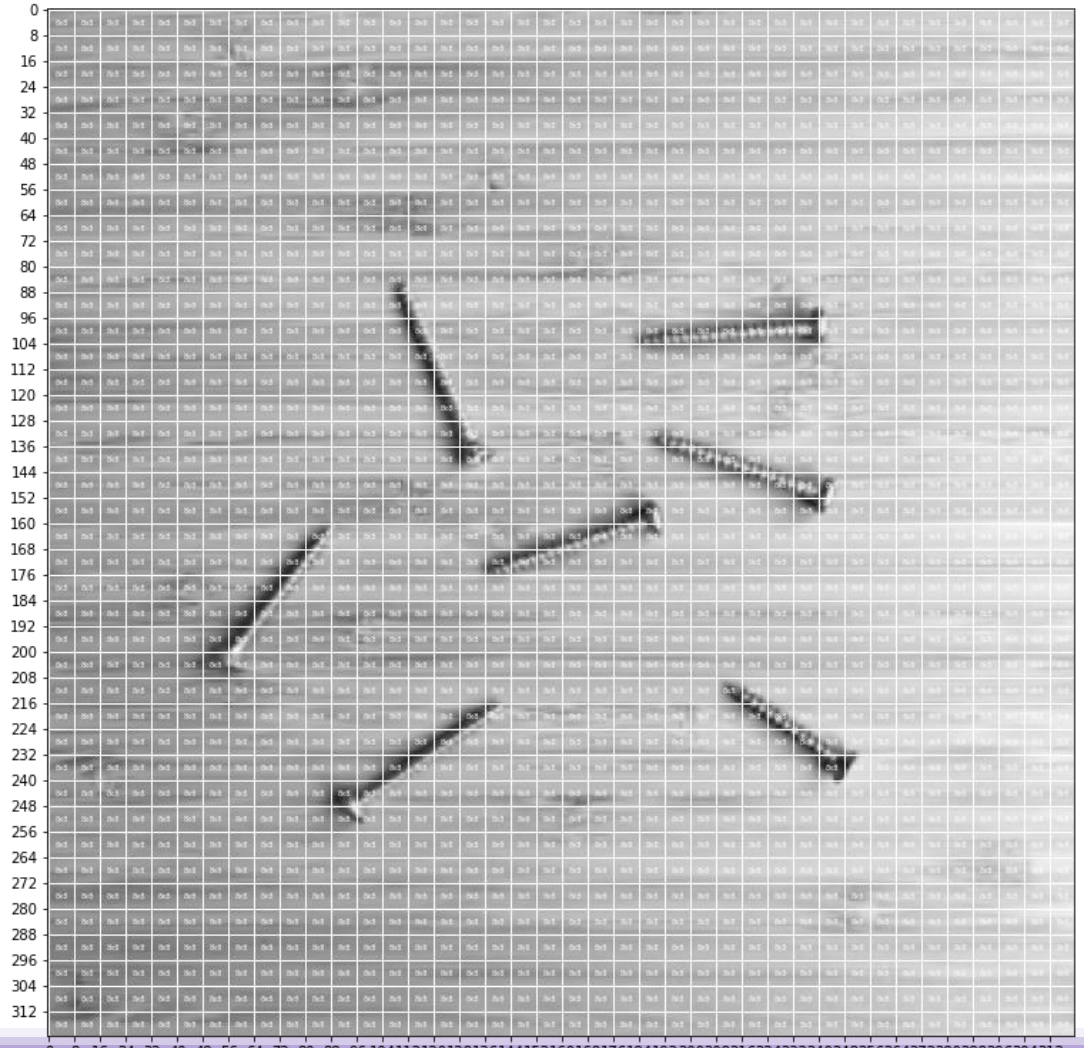
The question the model is trying to answer is:
“Are there faces in the image, where are they?”

Trained on centroid

when we run the model on an input image, we split this image into a grid and run the equivalent of image classification across all cells in the grid independently in parallel. By default the grid size is **8x8 pixels**, which means for a **96x96 image, the output will be 12x12.** For a 320x320 image, the output will be 40x40.



96x96 image split into a 12x12 grid

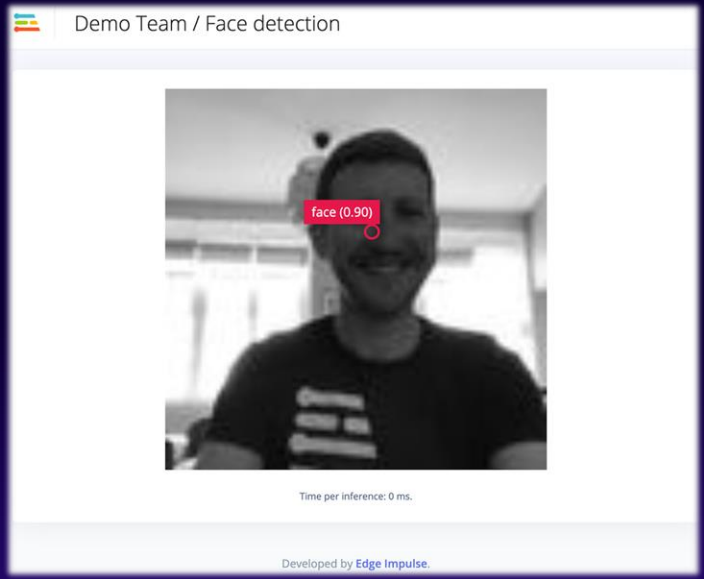

0 8 16 24 32 40 48 56 64 72 80 88 96 104 112 120 128 136 144 152 160 168 176 184 192 200 208 216 224 232 240 248 256 264 272 280 288 296 304 312
0 8 16 24 32 40 48 56 64 72 80 88 96 104 112 120 128 136 144 152 160 168 176 184 192 200 208 216 224 232 240 248 256 264 272 280 288 296 304 312

Trained on centroid

The main design decisions for **FOMO** are based on the idea that a lot of object detection problems don’t actually need the size of the objects but rather just their location in the frame.

Once we know the location of salient objects we can ask further questions such as "Is an object above/below another?" or "How many of these objects are in view?"

Based on this observation, classical bounding boxes are no longer needed. Instead, a detection based on the centroids of objects is enough.



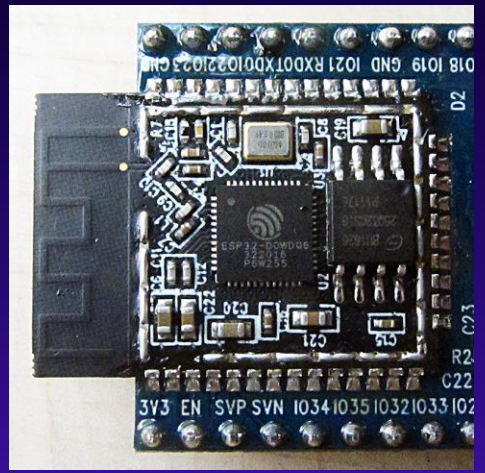
The question the model is trying to answer is:
“Are there faces in the image, where are they?”

Designed to be small and amazingly fast

One of the first goals, when we started to design FOMO, was to run object detection on microcontrollers where flash and RAM are most of the time very limited.

The smallest version of FOMO (96x96 grayscale input, MobileNetV2 0.05 alpha) runs in <100KB RAM and ~10 fps on a Cortex-M4F at 80MHz.

Core	Dual-core 32-bit MCU
Microprocessors	Two Xtensa 32-bit LX6 microprocessors: core 0 and core 1
CPU clock frequency	Adjustable from 80 MHz to 240 MHz
Low-power coprocessor	Yes, can be used instead of the CPU to save power

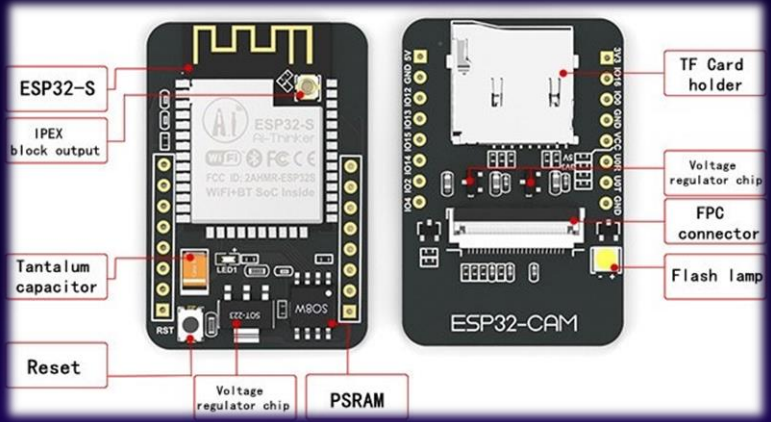


Limitations

Works better if the objects have a similar size

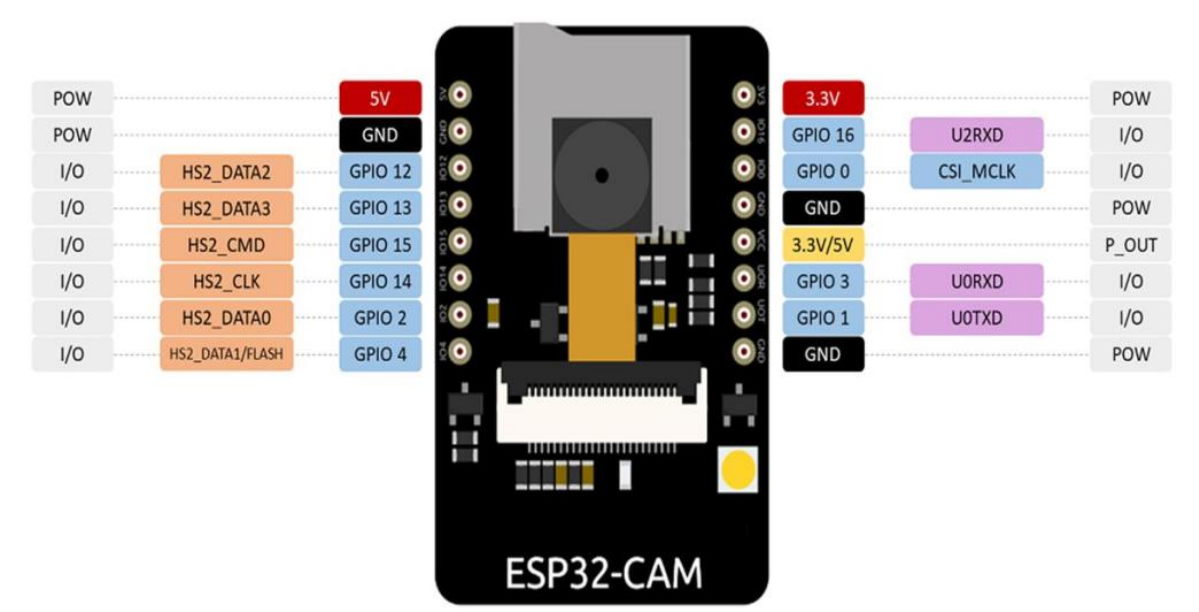
Objects shouldn’t be too close to each other.

ESP32_CAM


ESP32-S
IPEX block output
Tantalum capacitor
Reset
Voltage regulator chip
PSRAM
TF Card holder
Voltage regulator chip
FPC connector
Flash lamp
ESP32-CAM

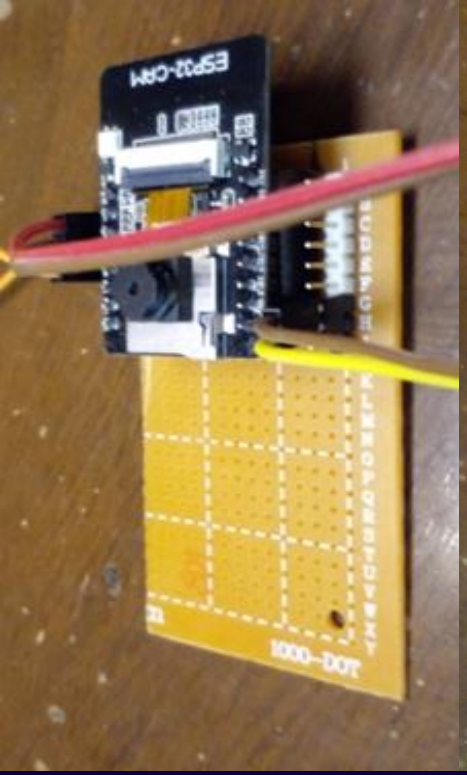

ESP32-CAM

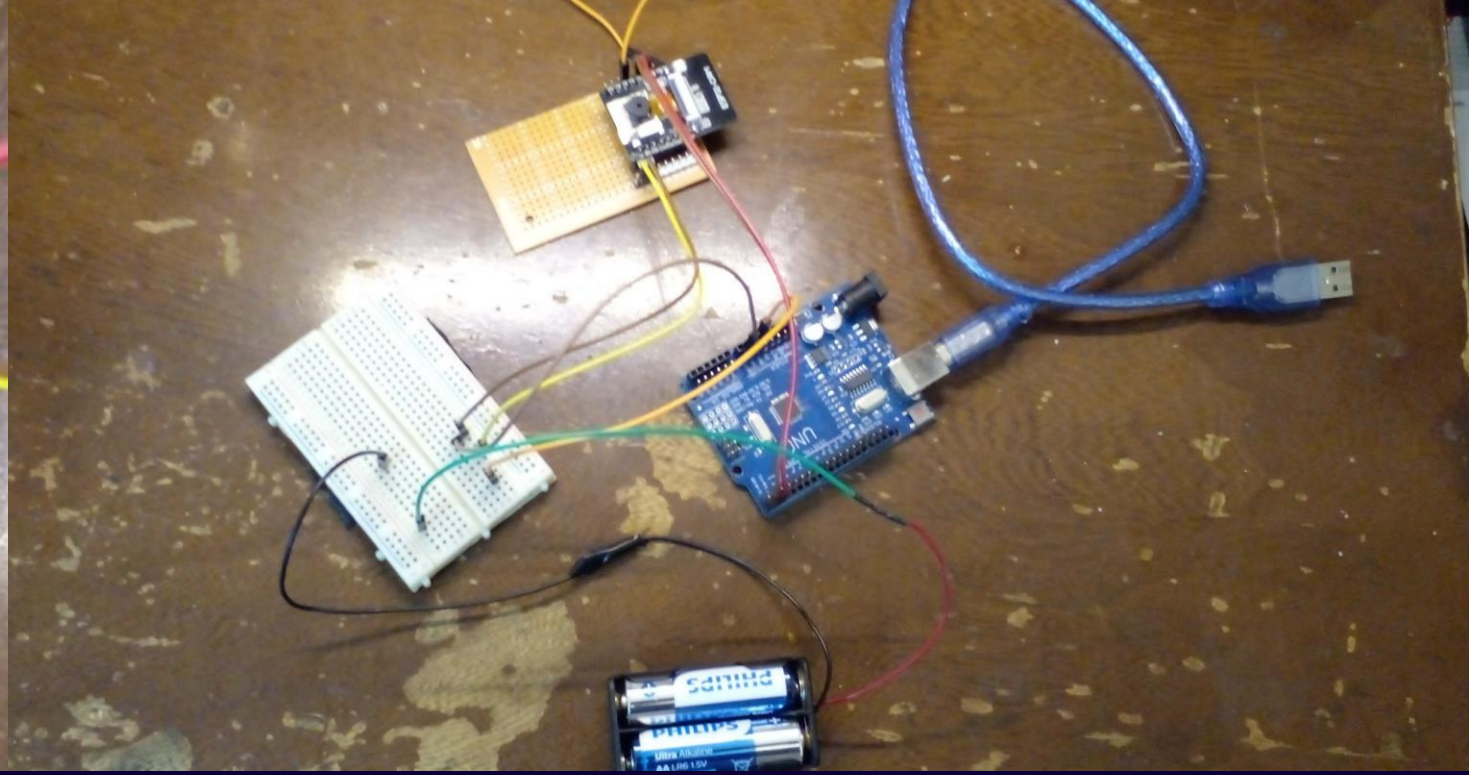
ESP32_CAM


POW
POW
I/O
I/O
I/O
I/O
I/O
I/O
HS2_DATA2
HS2_DATA3
HS2_CMD
HS2_CLK
HS2_DATA0
HS2_DATA1/FLASH
5V
GND
GPIO 12
GPIO 13
GPIO 15
GPIO 14
GPIO 2
GPIO 4
3.3V
GPIO 16
GPIO 0
GND
3.3V/5V
GPIO 3
GPIO 1
GND
U2RXD
CSI_MCLK
U0RXD
U0TXD
POW
I/O
I/O
POW
P_OUT
I/O
I/O
POW
ESP32-CAM

ESP32_CAM

- Product Name: ESP32-CAM.
- WiFi+Bluetooth module: ESP-32S.
- Camera Module: OV2640 2MP.
- Flash Light: LED Built-in on Board.
- Operating Voltage: 3.3/5 Vdc.
- Onboard TF card slot, supports up to 4G TF card for data storage .
- RAM: Internal 512KB + External 4MB PSRAM.
- Power consumption:
  - Flash off: 180mA@5V.
  - Flash on and brightness max: 310mA@5V.
  - Deep-Sleep: as low as 6mA@5V.
  - Modern-Sleep: as low as 20mA@5V.
  - Light-Sleep: as low as 6.7mA@5V
- Dimensions: 40.5mm x 27mm x 4.5mm


ESP32-CAM
1000-DOT


UNO
PHILIPS
Ultra Alkaline

Esp32 have the capability of connect to wifi and then stream on a local ip. You can enter the IP in your browser and watch the stream.

The vey first Picture I took at 3 or 4 PM after lots of misery

Toggle OV2640 settings

XCLK MHz 20 Set

Resolution QVGA(320x240)

Quality

Brightness

Contrast

Saturation

Special Effect

AWB

AWB Gain

WB Mode

AEC SENSOR

AEC DSP

AE Level

AGC

Gain Ceiling 2x 128x

BPC

WPC

Raw GMA

Lens Correction

H-Mirror

V-Flip

DCW (Downsize EN)

Color Bar

LED Intensity 0 255

Face Detection

Face Recognition

Get Still Stop Stream Enroll Face

UXGA(1600x1200)
SXGA(1280x1024)
HD(1280x720)
XGA(1024x768)
SVGA(800x600)
VGA(640x480)
HVGA(480x320)
CIF(400x296)
QVGA(320x240)
240x240
HQVGA(240x176)
QCIF(176x144)
QQVGA(160x120)
96x96

Toggle OV2640 settings

XCLK MHz 20 Set

Resolution QVGA(320x240)

Quality 4 63

Brightness -2 2

Contrast -2 2

Saturation -2 2

Special Effect No Effect

AWB

AWB Gain

WB Mode Auto

AEC SENSOR

AEC DSP

AE Level -2 2

AGC

Gain Ceiling 2x 128x

BPC

WPC

Raw GMA

Lens Correction

H-Mirror

V-Flip

DCW (Downsize EN)

Color Bar

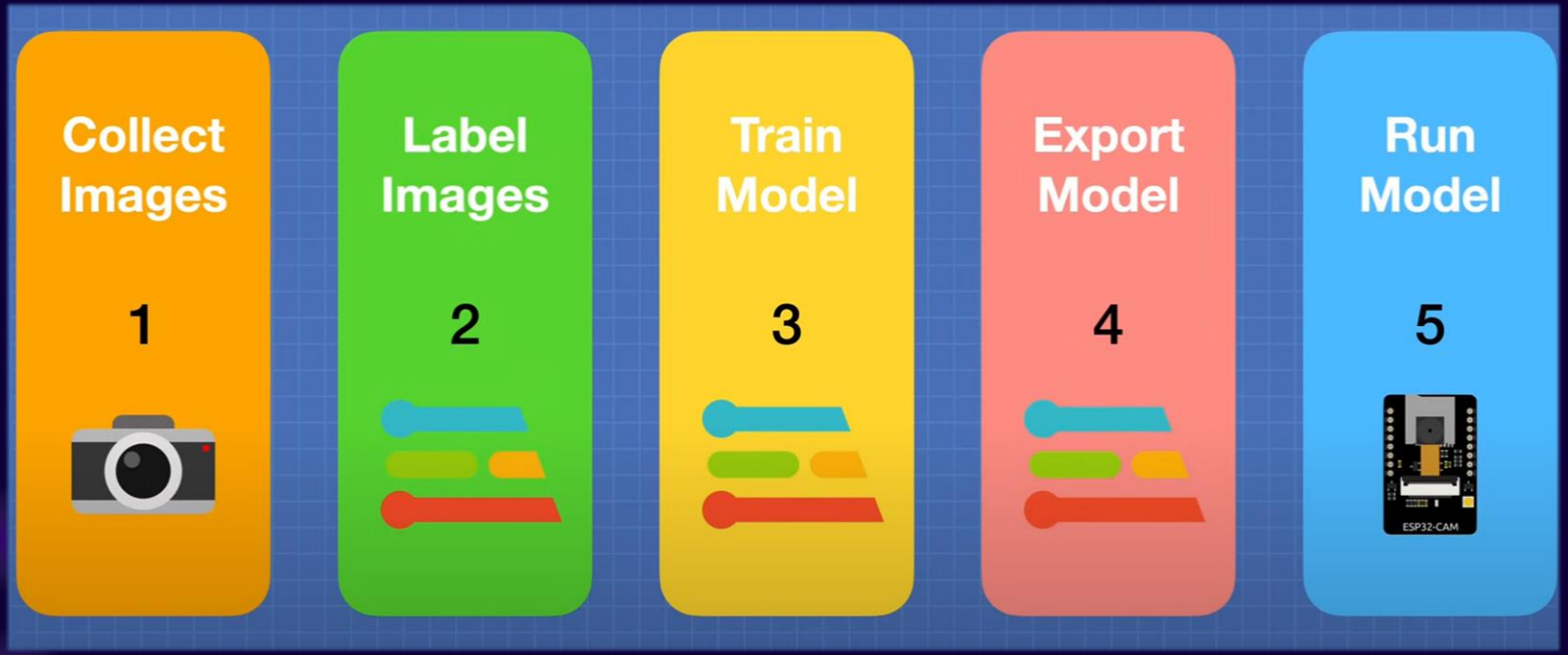
LED Intensity 0 255

Face Detection

Face Recognition

Get Still Stop Stream Enroll Face

Main Steps for Object detection using ESP32-CAM



EDGE IMPULSE

Valar Morghulis / Object-detection PERSONAL

Target: Espressif ESP-EYE...

VM

- Dashboard
- Devices
- Data acquisition
- Experiments
- EON Tuner
- Impulse #1
  - Create impulse
  - Image
  - Object detection

Upgrade Plan

Get access to higher job limits, collaborators and a full commercial license.

View plans

Dataset | Data sources | Labeling queue (0) | AI labeling

DATA COLLECTED
181 items

TRAIN / TEST SPLIT
75% / 25%

Dataset

Training (135) Test (46)

SAMPLE NAME	LABELS	ADDED
Scissors_1734177027309	Scissors	Dec 15 2024, 2...
Scissors_1734177018812	Scissors	Dec 15 2024, 2...
Scissors_1734177018575	Scissors	Dec 15 2024, 2...
Scissors_1734177018307	Scissors	Dec 15 2024, 2...
Scissors_1734177010809	Scissors	Dec 15 2024, 2...
Scissors_1734177003809	Scissors	Dec 15 2024, 2...
Scissors_1734177010566	Scissors	Dec 15 2024, 2...

Collect data

Connect a device to start building your dataset.

RAW DATA

Scissors_1734177027309

Metadata

Impulse #1

An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.

Image data

Input axes

image

Image width 96

Image height 96

Resize mode

Fit shortest

Add a processing block

Add a learning block

Output features

-

Save Impulse

Add a processing block

Did you know? You can bring your own DSP code.

DESCRIPTION	AUTHOR	RECOMMENDED	
Image OFFICIALLY SUPPORTED Preprocess and normalize image data, and optionally reduce the color depth.	Edge Impulse	★	Add
Flatten OFFICIALLY SUPPORTED Flatten an axis into a single value, useful for slow-moving averages like temperature data, in combination with other blocks.	Edge Impulse		Add
Audio (MFCC) OFFICIALLY SUPPORTED Extracts features from audio signals using Mel Frequency Cepstral Coefficients, great for human voice.	Edge Impulse		Add
Audio (MFE) OFFICIALLY SUPPORTED Extracts a spectrogram from audio signals using Mel-filterbank energy features, great for non-voice audio.	Edge Impulse		Add
Spectral Analysis OFFICIALLY SUPPORTED Great for analyzing repetitive motion, such as data from accelerometers. Extracts the frequency and power characteristics of a signal over time.	Edge Impulse		Add
Spectrogram OFFICIALLY SUPPORTED Extracts a spectrogram from audio or sensor data, great for non-voice audio or data with continuous frequencies.	Edge Impulse		Add

Add a learning block

Did you know? You can bring your own model in PyTorch, Keras or scikit-learn.

DESCRIPTION	AUTHOR	RECOMMENDED	
Object Detection (Images) OFFICIALLY SUPPORTED Fine tune a pre-trained object detection model on your data. Good performance even with relatively small image datasets.	Edge Impulse	★	Add
Object Detection (Images) - BrainChip Akida™ OFFICIALLY SUPPORTED Fine tune a pre-trained object detection model on your data. Good performance even with relatively small image datasets. Only works with BrainChip AKD1000 MINI PCIe board.	BrainChip	★	Add
Classification OFFICIALLY SUPPORTED Learns patterns from data, and can apply these to new data. Great for categorizing movement or recognizing audio.	Edge Impulse		Add
Transfer Learning (Images) OFFICIALLY SUPPORTED Fine tune a pre-trained image classification model on your data. Good performance even with relatively small image datasets.	Edge Impulse		Add
Regression OFFICIALLY SUPPORTED Learns patterns from data, and can apply these to new data. Great for predicting numeric continuous values.	Edge Impulse		Add
Transfer Learning (Keyword Spotting) OFFICIALLY SUPPORTED Fine tune a pre-trained keyword spotting model on your data. Good performance even with relatively small keyword datasets.	Edge Impulse		Add

Parameters | Generate features

Raw data

Show: All labels | Scissors_1734177027309 (Scissors

Raw features

0x667268, 0x667268, 0x667268, 0x667268, 0x667268, 0x667268, 0x68746a, 0x69756b, 0x69756b, 0x6b776d, 0x6d796f, 0x6d796f, 0x6d7…

Parameters

Image

Color depth: Grayscale

Save parameters

DSP result

Image

Processed features

0.4285, 0.4285, 0.4285, 0.4285, 0.4285, 0.4285, 0.4364, 0.4403, 0.4403, 0.4481, 0.4560, 0.4560, 0.4560, 0.4481, 0.4481, 0.448…

On-device performance

PROCESSING TIME
15 ms.

PEAK RAM USAGE
4 KB

Parameters | Generate features

Training set

Data in training set	135 items
Classes	2 (Apple, Scissors)

Generate features

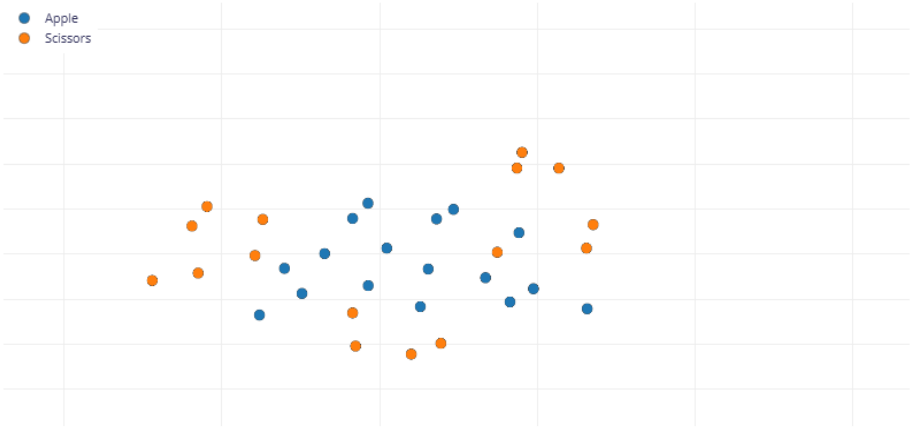
Feature generation output

(0)

```
Job started at 17 Dec 2024 19:06:48
Reducing dimensions for visualizations...
UMAP( verbose=True)
Tue Dec 17 19:07:00 2024 Construct fuzzy simplicial set
Tue Dec 17 19:07:00 2024 Finding Nearest Neighbors
Tue Dec 17 19:07:02 2024 Finished Nearest Neighbor Search
Tue Dec 17 19:07:04 2024 Construct embedding
Epochs completed: 100% 500/500 [00:02<00:00, 222.86it/s]
Tue Dec 17 19:07:06 2024 Finished embedding
Writing output files...
Writing output files OK
Reducing dimensions for visualizations OK (took 6011ms.)

Job completed (success)
```

Feature explorer



On-device performance

PROCESSING TIME
15 ms.

PEAK RAM USAGE
4 KB

Neural Network settings

Training settings

Number of training cycles: 60

Use learned optimizer: ☐

Learning rate: 0.001

Training processor: CPU

Data augmentation: ☑

Advanced training settings

Neural network architecture

Input layer (9,216 features)

FOMO (Faster Objects, More Objects) MobileNetV2 0.1

Choose a different model

Output layer (2 classes)

Save & train

Training output

(0)

Model

Model version: Quantized (int8)

Last training performance (validation set)

F1 SCORE
83.3%

Confusion matrix (validation set)

	BACKGROUND	APPLE	SCISSORS
BACKGROUND	100%	0%	0%
APPLE	25%	75%	0%
SCISSORS	33.3%	0%	66.7%
F1 SCORE	1.00	0.86	0.80

Metrics (validation set)

METRIC	VALUE
Precision (non-background)	1.00
Recall (non-background)	0.71
F1 Score (non-background)	0.83

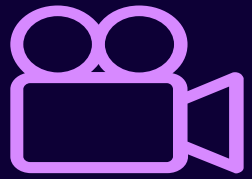
On-device performance

Engine: EON™ Compiler

INFERENCING TIME
937 ms.

PEAK RAM USAGE
235.5K

FLASH USAGE
65.0K



Video

Resources

www.datacamp.com

www.studio.edgeimpulse.com

www.edgeimpulse.com

www.youtube.com

www.docs.edgeimpulse.com

www.geeksforgeeks.org

www.geeksforgeeks.org

www.geeksforgeeks.org

www.geeksforgeeks.org