life.augmented

ST AI Solutions on computer vision

Asia Pac Artificial Intelligence Competence Center

Di LI

Agenda

1 Overview ST solutions for CV

2 Network optimization and deployment strategy on STM32

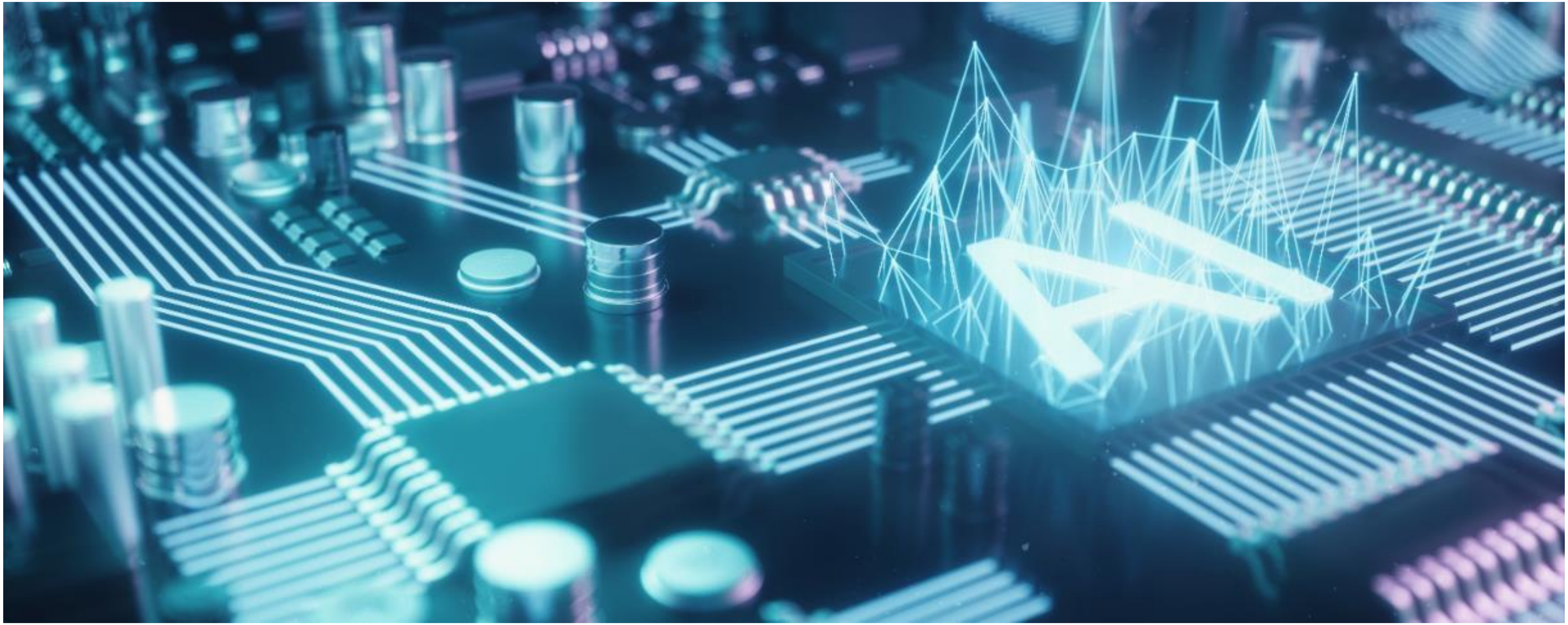
3 Introduction to FP-AI-VISION1 function pack

life.augmented

Overview ST solutions for CV

life.augmented

This is where AI opens new horizons for embedded design!



life.augmented

Product development new paradigm

From rule-based engineering to data-driven engineering

Standard programming

Handcrafted rules based on experience

- Requires digital signal processing skills
- Manual feature extraction?
- Need to rewrite if environment evolves

Machine Learning

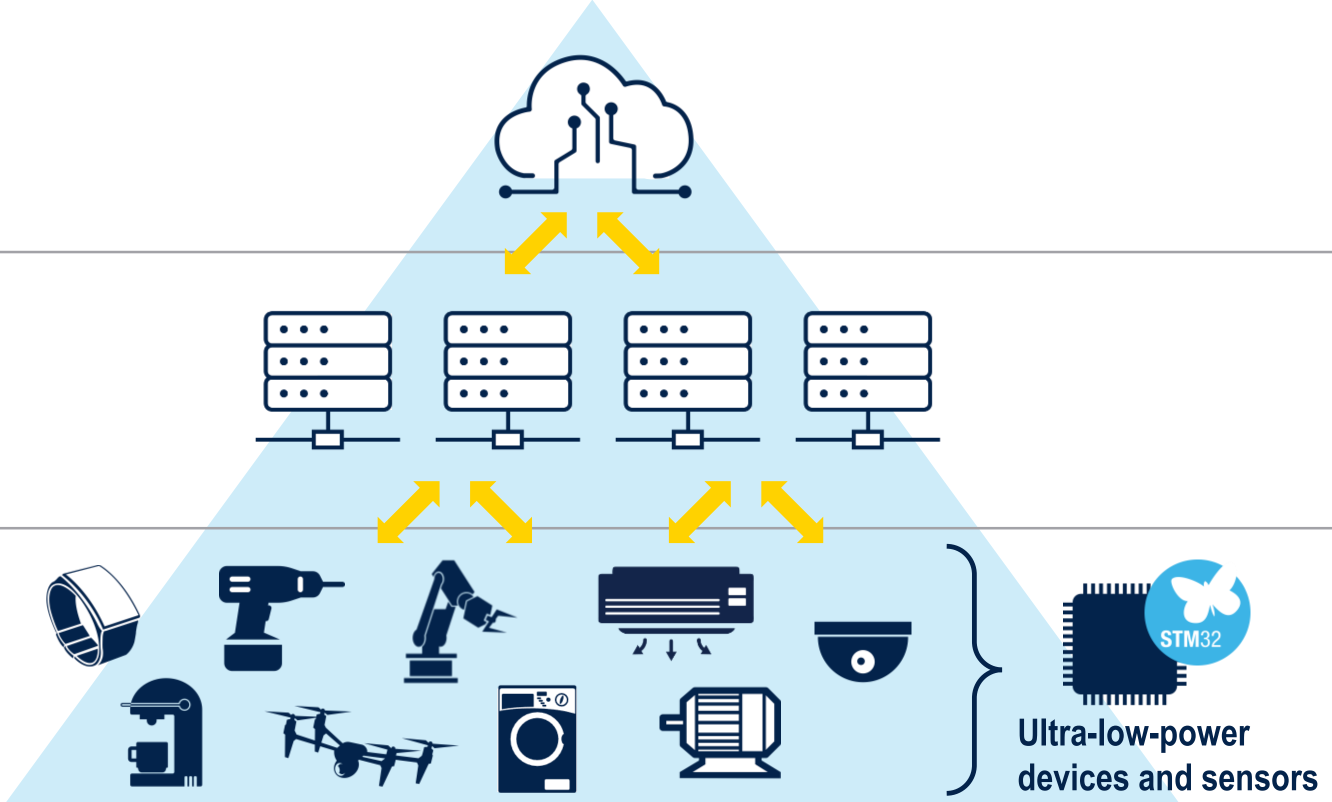
Rules learnt from real-world data

- Generate code from real-world observations
- Automated feature extraction?
- Re-learn from data if environment evolves

life.augmented

Distributed Artificial Intelligence approach

Leverage billions of devices at the Edge!



Thousands

Data Center Cloud

Analytics, storage, compute

Millions

Edge Nodes

IoT gateways, micro datacenters

100 Billions

Edge Things

Real time, local processing

life.augmented

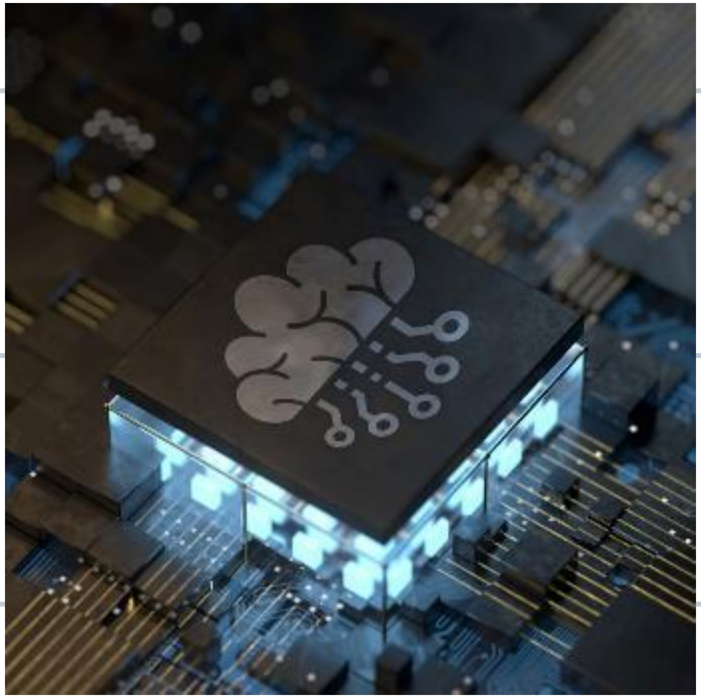
Addressing the challenges of your IoT products

Artificial Intelligence close to data acquisition brings several benefits

Ultra-low latency
Real-time applications

More reliability

Security of data
No sharing in the cloud



Privacy by design
GDPR compliant

Sustainable on energy
Low-power consumption

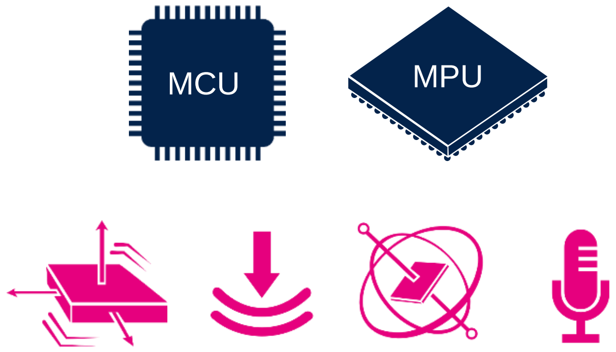
Better user experience

Embedded AI technology trend

“Global Shipments of Deep Edge AI Devices to Reach 2.5 Billion by 2030”

Source: [ABI Research]

AI technologies are now embedded inside end devices (MPU, MCU and sensors).



Growing community and ecosystem of **Deep Edge AI** technologies focusing on standalone, low-power and cost-efficient embedded solutions.

AI development workflow – STM32Cube.AI

1 Data preparation

- Data acquisition
- Data processing

Data logging tools

2 Model development

- Model selection and training
- Model testing

scikit learn, Keras, TensorFlow Lite, PyTorch, ONNX

3 Model Implementation

- Model library creation
- Model inference

Edge AI toolkit

STM32Cube.AI

STM32

Functions served by Neural Networks

Classification

DOG

Object Detection

DOG, PERSON

Image Segmentation

DOG, PERSON

life.augmented

Computer vision boards for AI

STM32**L4**	L4 / L4+ Discovery

[32L4R9IDISCOVERY](#)

Flash: 2MB
RAM: 640kB
LCD: 1.2" 390x390 px capacitive touch round display panel
Camera: USB OTG FS

STM32**H7**	Discovery

[STM32H747I-DISCO](#)

Flash: 2MB
RAM: 1MB
LCD: 4” capacitive touch
Camera: F4DIS-CAM or VG5661

STM32**MP1**	MP1 Discovery and 96Boards

[STM32MP157C-DK2](#)

Flash: uSD card
RAM: 512 MB
LCD: 4" TFT 480×800 pixels
Camera: USB OTG HS
1 Gb Ethernet, 802.11b/g/n, BLE4.1

[Arrow Avenger96](#)

Flash: 8GB eMMC + 2MB QSPI NOR
RAM: 1GB DDR3
LCD: HDMI1.4 WXGA
1Gb Ethernet, 802.11a/b/g/n/ac, BLE4.2

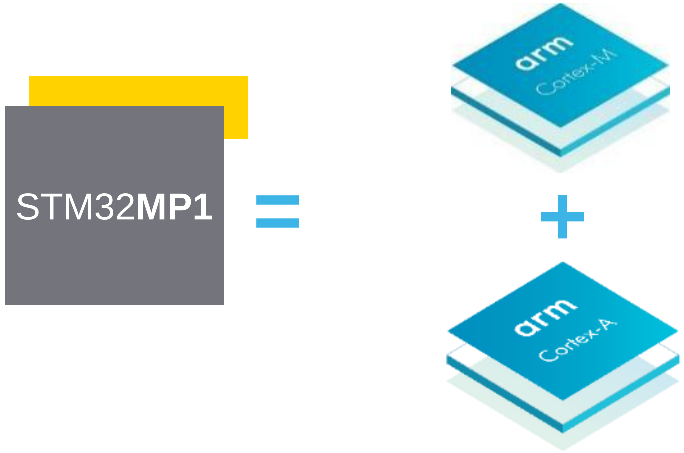
[D3 CAMERA MEZZ OV5640](#)

Camera: [OV5640](#) image sensor
RGB 15 FPS 5Mpx or 30 FPS 1080p HD
MIPI CSI2.0

life.augmented

STM32MP1 microprocessor Augmented intelligence

Cortex-M4
Realtime OS
STM32CubeMX



Dual Cortex-A7
OpenSTLinux Distribution
STM32CubeMx

STM32 Cube.AI

X-LINUX-AI-CV
support for
TensorFlow Lite
&
OpenCV

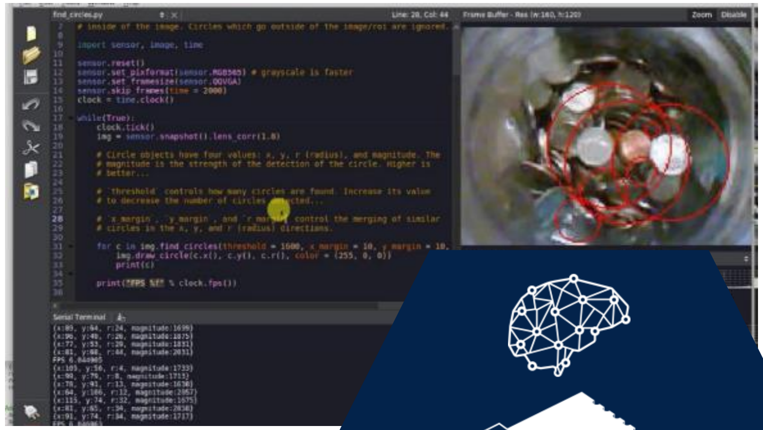
- STM32Cube.AI to convert pre-trained NNs for the Cortex-M4 core
- TensorFlow Lite STM32MP1 support up streamed for native NN inferences support on the dual Cortex-A side

life.augmented

OpenMV integration
Fast machine vision prototyping

Configure Machine Vision in real-time over USB in Python

Run and validate optimized Neural Network



STM32 Cube.AI

OpenMV CAM
Running MicroPython over STM32

https://github.com/openmv/openmv

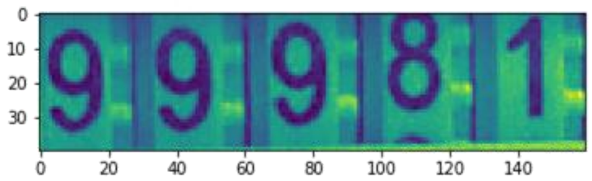
Aftermarket wireless digit reader for metering

Equip meters with aftermarket Wireless & Low power reader

Use case

- Equip meter with ad-on SPI low-cost camera
- Boost ROI avoiding onsite visit implementing long range wireless reader
- Electrical, gaz, water meters supported
- Reader lifetime : 2 years on battery at ? read per hour
- Reading sent over LoRaWAN

STM32WL55JC
ov2640



Demo overview

- Input : QQVGA @4fps
- Proprietary Neural Network
    - Accuracy : 98%
    - Inference 84 ms / digit
    - Ram : 21KiB
    - Flash 20KiB
    - Trained on a private dataset
    - LoRaWan Stack on LoRa SoC
    - Ram 6KB
    - Flash 65KB

STM32 Cube.AI

Neural Network on STM32WL

life.augmented

Aftermarket wireless digit

Equip meters with aftermarket Wireless & Low power reader

Demo setup

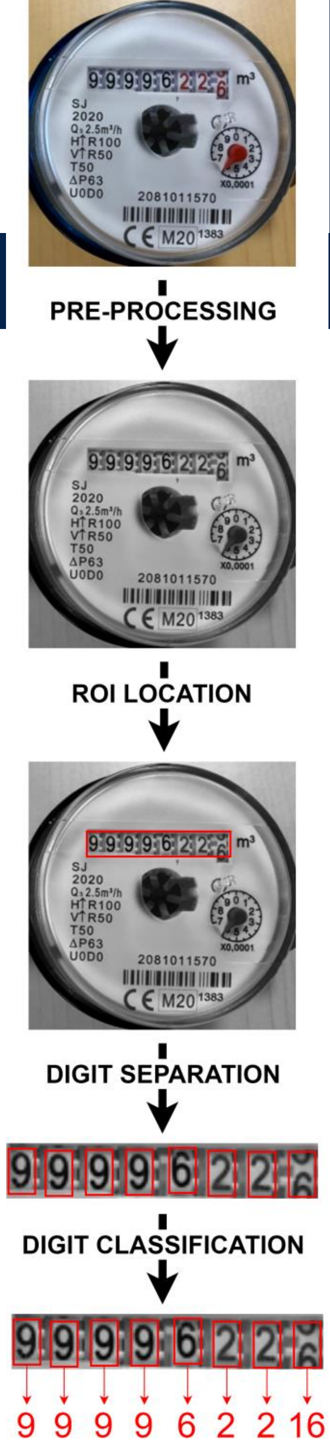
B-L462E-CELL1 board with LBAD0ZZ1SE module from Murata:

- STM32L462RE
  512 KB Flash, 160KB RAM, 80 MHz
- eSIM (ST4SIM-200M),
- LTE Cat M/NBIoT modem

Arducam mini 5MP plus

Neural Networks

- **ROI NN detection**
  - Input : 240x240
  - Quantized CNN
  - 148 KB Flash / 57 KB RAM
  - Inference time 0.3 s
- **Digits NN recognition**
  - Input : 24x140
  - Quantized CNN
  - 67 KB Flash / 66 KB RAM
  - Inference time 0.9 s
  - Output: 8 digit including half on latest digit

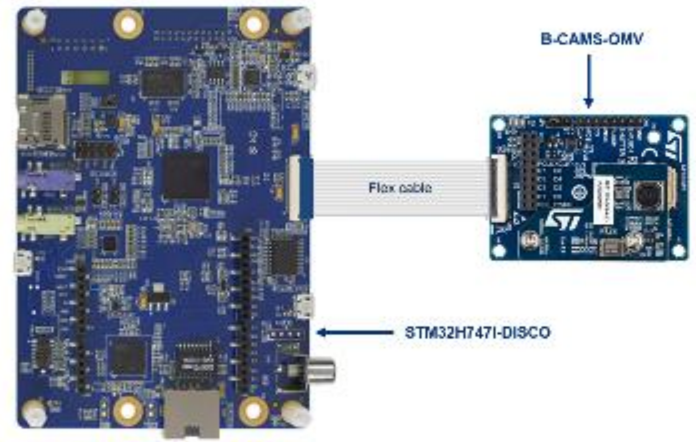


STM32 Cube.AI

Neural Network on **STM32L4**

life.augmented

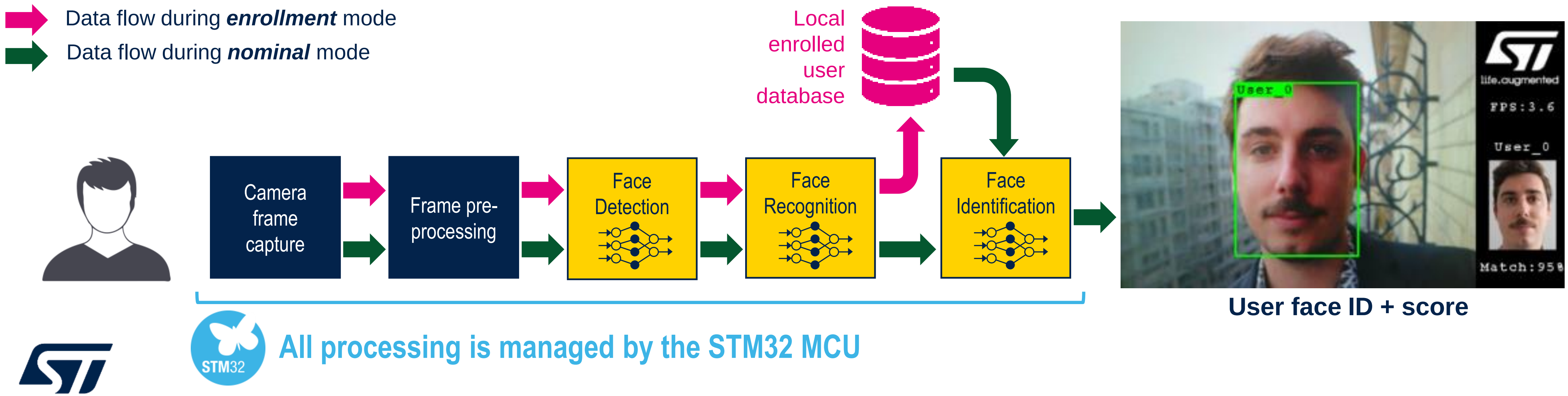
Face recognition



Recognize one or multiple human faces within a camera captured frame.

Face recognition at the edge is becoming more and more popular because of following advantages:

- Privacy issue : no loss of privacy thanks to local image processing
- Extremely low latency
- Low power consumption



Object detection on STM32MP1 MPU

Advanced object detection among 90 different objects using TensorFlow Lite on STM32MP1

Use cases

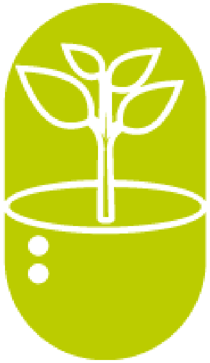
- Detect object and its position.
- 90 objects can be detected
- Object detection is performed in real time for fast interaction with user
- Requires only a low-resolution camera

Arrow Avenger96 or STM32MP1 DK2
DCMI Camera@30fps or USB Camera
Display

Demo overview

- TensorFlowLite integrated via C++ runtime implementation on STM32MP1 dual-core A7
- COCO SSD MobileNet v1: 90 objects
- CPU load balanced on the 2 cores
- Processing time: 1.1 FPS

Identify and locate potential instances of plant disease



STM32 Cube.AI

Neural Network on **STM32MP1**

life.augmented

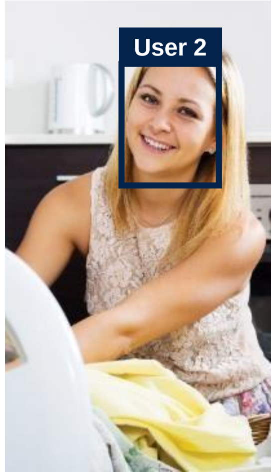
FP-AI-VISION1

Computer Vision Software for STM32

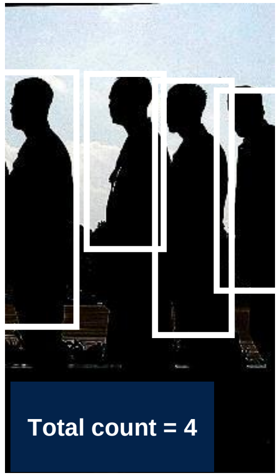
Add AI computer vision to your STM32 product for new features and add-on services

Person presence detection

Object classification



Face Recognition



People Counting

Keras TensorFlowLite ONNX
PyTorch And more

STM32 Cube.AI ↔ STM32 CubeMX

Optimized NN files
STM32.AI lib

Customer application
STM32.Vision lib

STM32 run-time

life.augmented

Making Edge AI possible with all STM32 portfolio

STM32Cube.AI is compatible with all STM32 series

LONGEVITY 10 YEARS COMMITMENT

MPU							**STM32MP1** 4158 CoreMark Up to 800 MHz Cortex-A7 209 MHz Cortex-M4
High Perf MCUs			**STM32F2** Up to 398 CoreMark 120 MHz Cortex-M3	**STM32F4** Up to 608 CoreMark 180 MHz Cortex-M4	**STM32F7** 1082 CoreMark 216 MHz Cortex-M7	**STM32H7** Up to 3224 CoreMark Up to 550 MHz Cortex -M7 240 MHz Cortex -M4	
Mainstream MCUs	**STM32F0** 106 CoreMark 48 MHz Cortex-M0	**STM32G0** 142 CoreMark 64 MHz Cortex-M0+	**STM32F1** 177 CoreMark 72 MHz Cortex-M3				
Mainstream MCUs (*Mixed-signal MCUs*)			**STM32F3** 245 CoreMark 72 MHz Cortex-M4	**STM32G4** 569 CoreMark 170 MHz Cortex-M4			
Ultra-low Power MCUs	**STM32L0** 75 CoreMark 32 MHz Cortex-M0+	**STM32L1** 93 CoreMark 32 MHz Cortex-M3	**STM32L4** 273 CoreMark 80 MHz Cortex-M4	**STM32L4+** 409 CoreMark 120 MHz Cortex-M4	**STM32L5** 443 CoreMark 110 MHz Cortex-M33	**STM32U5** 651 CoreMark 160 MHz Cortex-M33	
Wireless MCUs			**STM32WL** 162 CoreMark 48 MHz Cortex-M4 48 MHz Cortex-M0+	**STM32WB** 216 CoreMark 64 MHz Cortex-M4 32 MHz Cortex-M0+ ●			

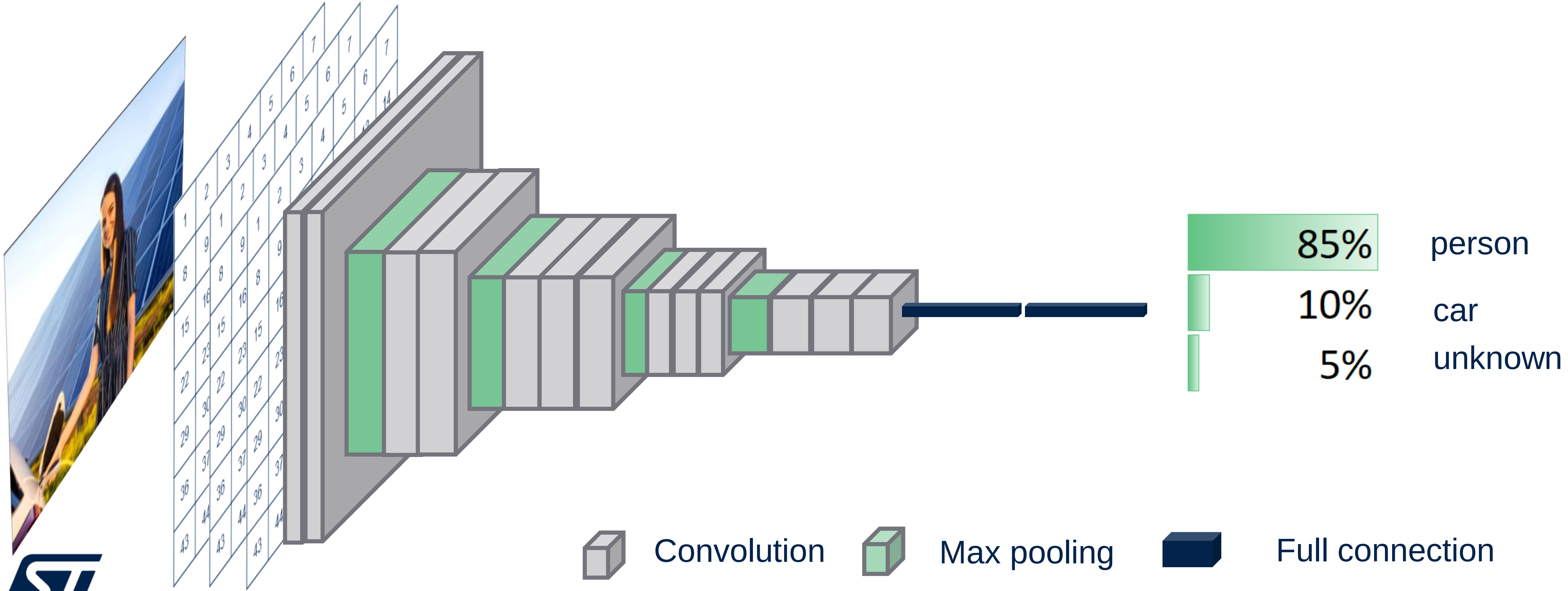
■ Latest product generation   ● Radio co-processor only

Network optimization and deployment strategy on STM32

life.augmented

Image classification model

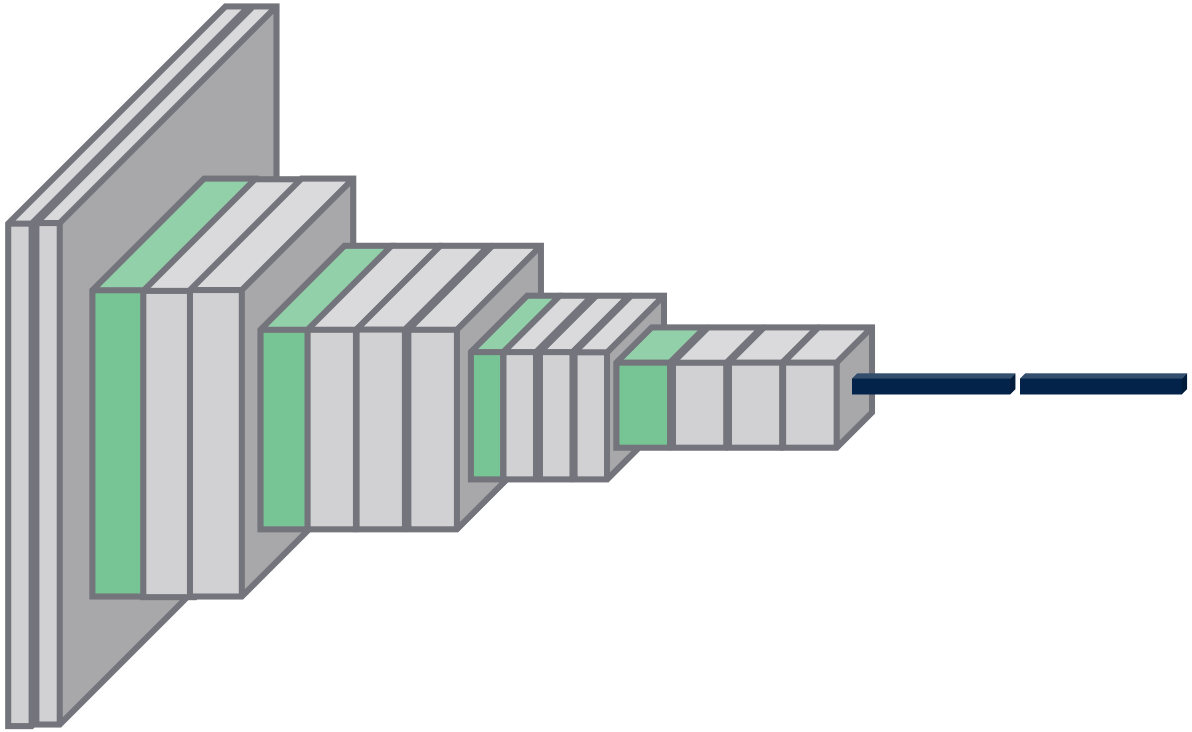
- The following is the classical architecture of convolutional neural network, VGG-16
- This structure is mainly composed of convolutional, pooling and fully-connected layers
- For the model of image classification, after the images are fed into the convolutional neural network, the network will output an array, each item of which is the probability of the corresponding class



VGG is too heavy!!

- The classic convolutional neural network VGG-16 is too bulky and heavy for embedded devices such as stm32. Because stm32 and other embedded devices RAM and flash resources are relatively limited.

VGG-16 structure



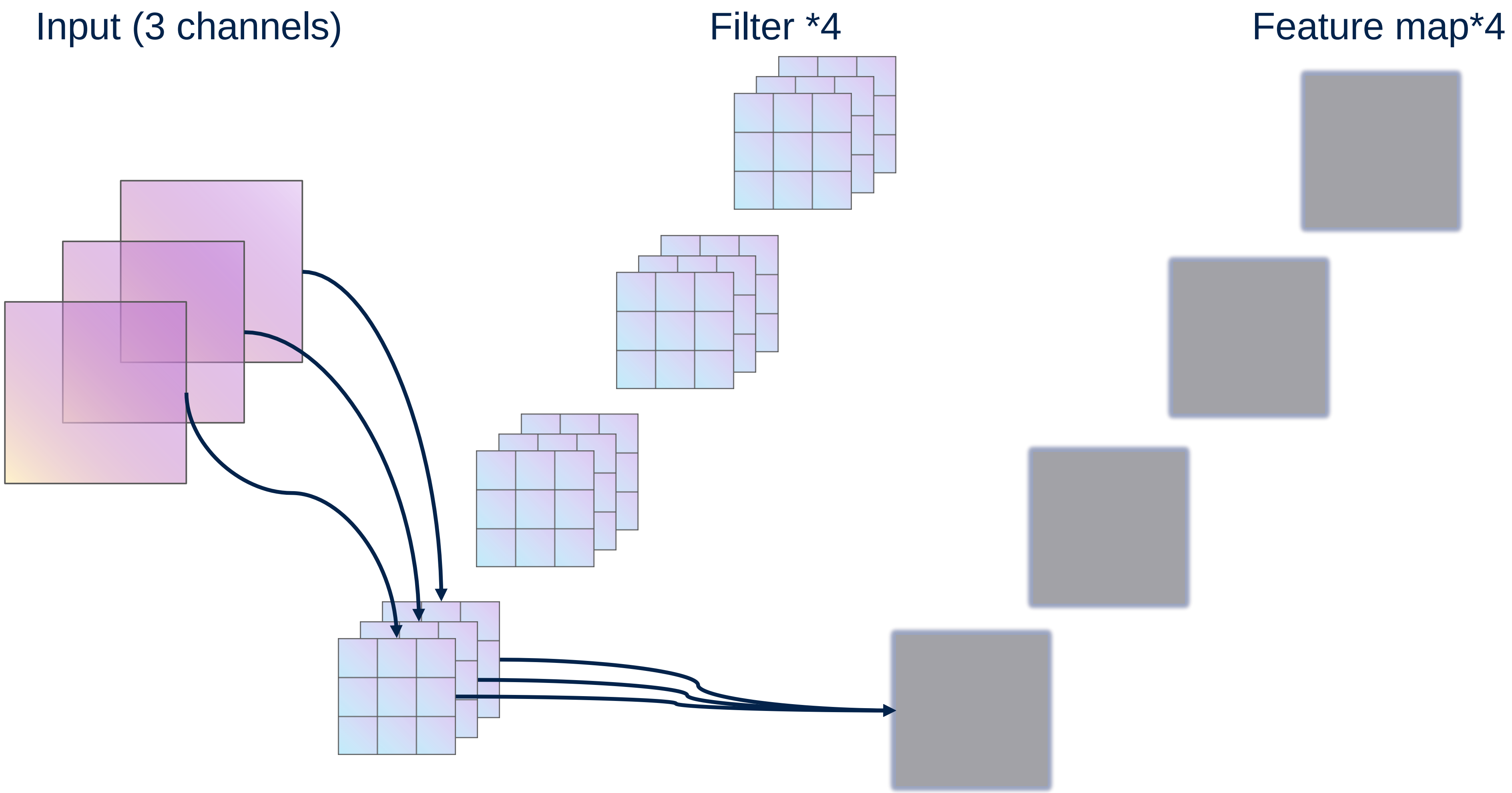
VGG-16 parameters

INPUT: [224x224x3] weights: 0
CONV3-64: [224x224x64] 】weights: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] weights: (3*3*64)*64 = 36,864
POOL2: [112x112x64] weights: 0
CONV3-128: [112x112x128] weights: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128weights: (3*3*128)*128 = 147,456
POOL2: [56x56x128] weights: 0
CONV3-256: [56x56x256] weights: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] weights: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] weights: (3*3*256)*256 = 589,824
POOL2: [28x28x256] weights: 0
CONV3-512: [28x28x512] weights: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] weights: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] weights: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] weights: 0
CONV3-512: [14x14x512] weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] weights: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] weights: 0
FC: [1x1x4096] weights: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] weights: 4096*4096 = 16,777,216
FC: [1x1x1000] weights: 4096*1000 = 4,096,000

TOTAL params: 138M parameters

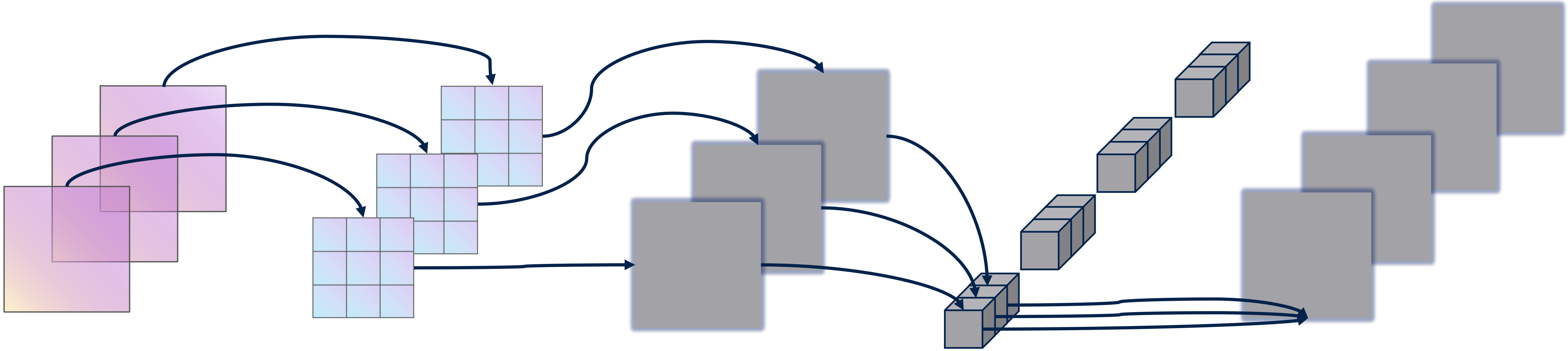
Why separable convolution

- Parameters required for classical convolution calculation
- N_std = 4 × 3 × 3 × 3 = 108



Why separable convolution

❑ Parameters required for classical convolution calculation



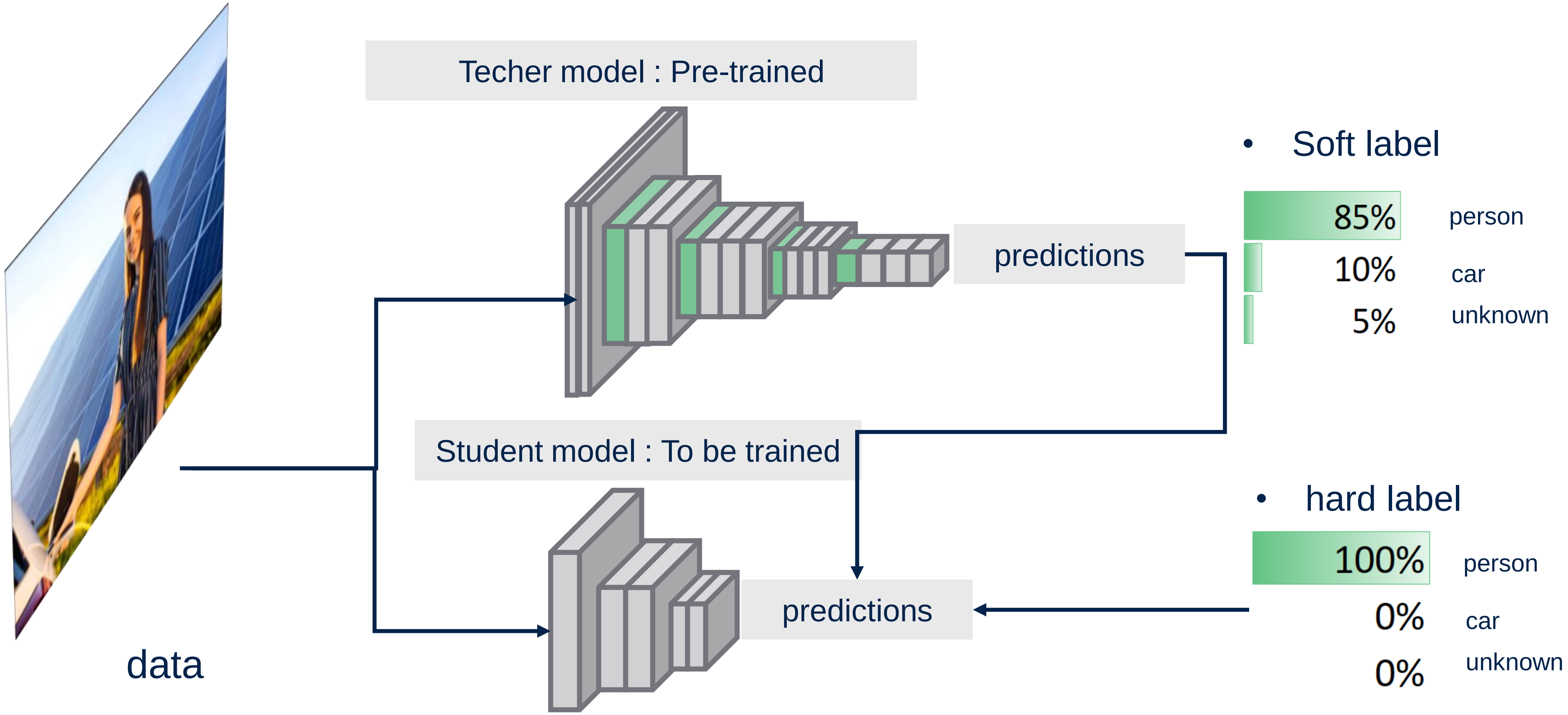
N_depthwise = 3 × 3 × 3 = 27

N_pointwise = 1 × 1 × 3 × 4 = 12

N_separable = N_depthwise + N_pointwise = 39

Knowledge distillation

- Knowledge distillation, which can transfer knowledge from one network to another. This is done by first training a TEACHER network, and then using the output of this TEACHER network and the true labels of the data to train the STUDENT network. Knowledge distillation can be used to transform a network from a large network into a small network and retain performance close to that of the large network.



Quantization

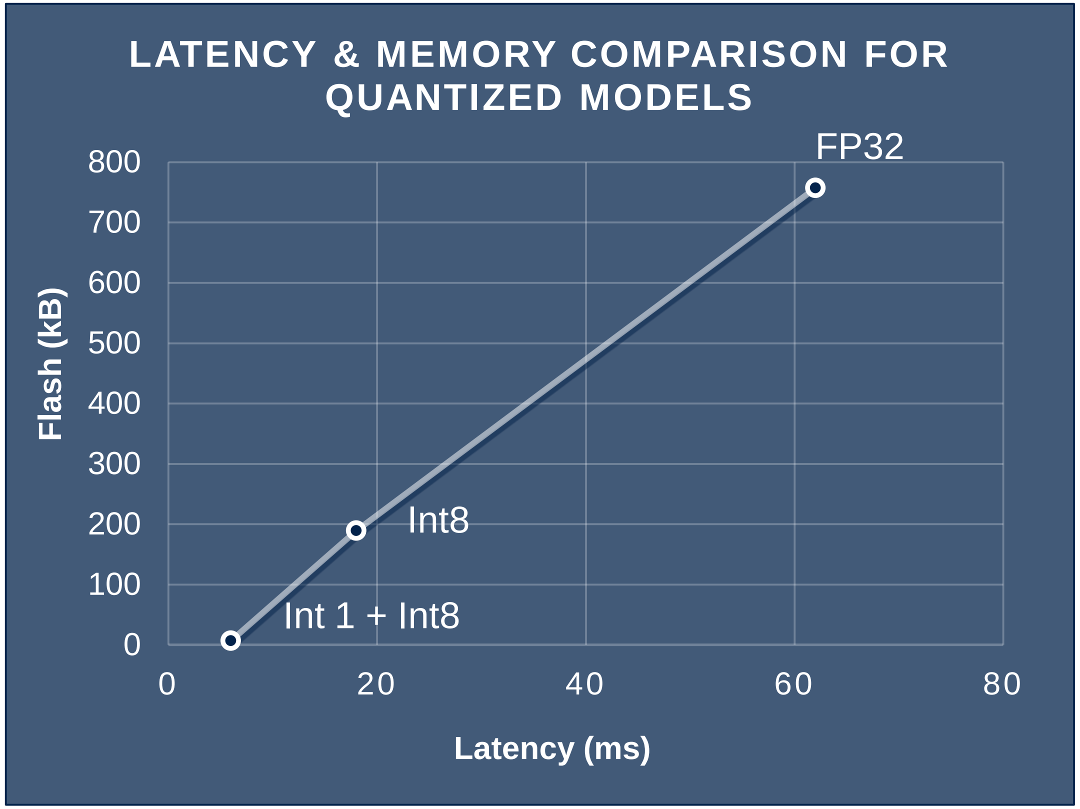
- The models are quantized. Quantization consist in converting floating-point (32b) model to fixed-point (8b) model
  - To reduce model size (size of the requested memory to store the weights). Up to x4
  - To reduce peak memory usage (size of the activations memory buffer). Up to x4
  - To improve latency (runtime is based on the integer operator implementations and uses the STM32 DSP instructions). Consequently, inference time and power consumption are improved.
  - With minimal loss of accuracy. Network size/complexity dependent

Model	Format	Accuracy (top 1)	Inference time (ms)	Weight memory Flash (KiB)	Activation footprint RAM (KiB)
Image Classification	Quantized	85 %	70	77	38
Image Classification	Float	87 %	212	311	132
%	float vs quantized		+ 203 %	+ 304 %	+ 247 %
Visual Wake Word	Quantized	85.2 %	58	214	37
Visual Wake Word	Float	85.4 %	190	824	150
%	float vs quantized		+ 228 %	+ 285 %	+ 305 %

life.augmented

Quantized model support

Simply use quantized networks to reduce memory footprint and inference time



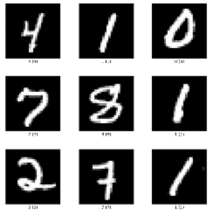
STM32Cube.AI support quantized Neural Network models with **all parameter formats**:

- FP32
- Int8
- Mixed binary Int1 to Int8 (Qkeras*, Larq.dev*)

**Please contact edge.ai@st.com to request the relevant version of STM32Cube.AI*

HW Target: NUCLEO-STM32H743ZI2
Model: Low complexity handwritten digit reading
Freq: 480 MHz
Accuracy: >97% for all quantized models

Tested database: MNIST dataset



MNIST dataset

life.augmented

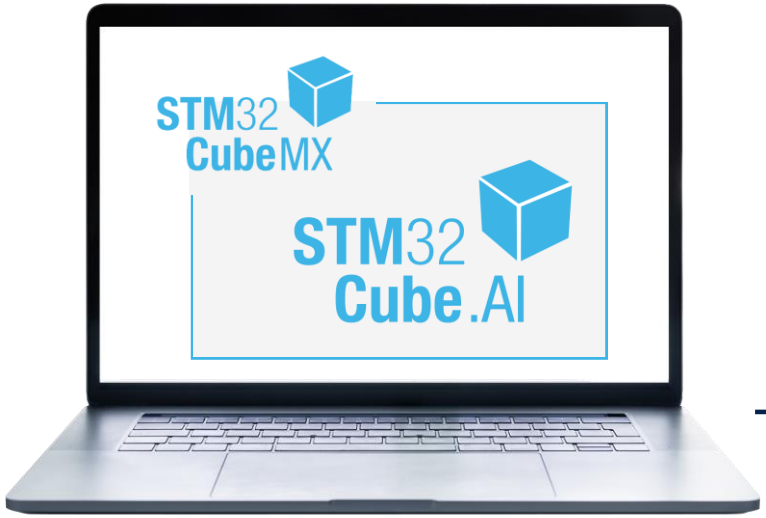
A tool to seamlessly integrate AI in your projects

Machine Learning

scikit learn
ONNX

Deep Learning

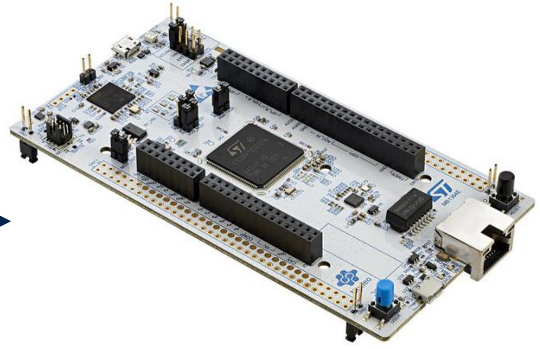
MATLAB
PyTorch
TensorFlow Lite
Keras
ONNX



Select MCU & upload your model

Optimize and validate

Generate project and deploy



HIGH PERFORMANCE EDGE AI PRODUCT

Layers / Frameworks support

- Find out all the **layers officially supported by Cube.AI** in the following repository
  - C:\Users\<user name>\STM32Cube\Repository\Packs\STMicroelectronics\X-CUBE-AI\7.2.0\Documentation

STM32 Cube.AI 7.2.0 X-CUBE-AI Documentation

[Index]

Embedded documentation (release note)
License
External resources

X-CUBE-AI Documentation

X-CUBE-AI Expansion Package

AI PLATFORM r7.2.0 (Embedded Inference Client API 1.2.0)
Command Line Interface r1.6.0

Embedded documentation (release note)

User Guides	Advanced features
Installation	Relocatable binary network support
API Breaking change	Keras stateful LSTM/GRU support
Command Line Interface	Keras Lambda/custom layer support
Embedded inference client API	Platform Observer API
Evaluation report and metrics	STM32 CRC IP as shared resources
Quantized model and quantize command	TensorFlow lite for micro-controller support

HowTo	Supported DL/ML frameworks
How to use USB-CDC driver for validation	Embedded Python modules - *Deep-Learning framework versions*
How to run locally a c-model	Keras toolbox
How to upgrade a STM32 project	TensorFlow Lite toolbox
	ONNX toolbox
	Machine Learning support (ONNX-ML operators)
	Deep Quantized Neural Network support (NEW)

life.augmented

Keras

- In *Keras* we support the Tensorflow backend with channels-last dimension ordering. Keras 2.0 up to version 2.3.1 is supported, while networks defined in Keras 1.x are not officially supported.
- Model may be loaded from a single file with model and weights (.h5, .hdf5) or from the model configuration and weight in separate files. In the latter case, the weights are loaded from a HDF5 file (.h5, .hdf5) and model configuration is loaded from a text file, either JSON (.json) or YAML (.yml, .yaml).

Keras operators supported by cube.AI

Dense	Activation	Flatten	Reshape	InputLayer	Permute	RepeatVector	Conv1D	Conv2D
SeparableConv1D	SeparableConv2D	DepthwiseConv1D	DepthwiseConv2D	Conv2DTranspose	Cropping1D	Cropping2D	Upsampling1D	Upsampling2D
ZeroPadding1D	ZeroPadding2D	MaxPooling1D	MaxPooling2D	AveragePooling1D	AveragePooling2D	GlobalMaxPooling1D	LSTM	GRU
ReLU	Softmax	BatchNormalization	Bidirectional	Dropout	GaussianDropout	Concatenate	ActivityRegularization	SpatialDropout1D

life.augmented

Tensorflow Lite

- *Tensorflow Lite* is the format used to deploy neural network models on mobile platforms. Cube.Ai converts the bytestream (.tflite files) to C code; a number of operators from the *supported operator* list are handled and quantized models are partially supported.

Tensorflow Lite operators supported by cube.AI								
AVERAGE_POOL_2D	MAX_POOL_2D	CONCATENATION	CONV_2D	TRANSPOSE_CONV	DEPTHWISE_CONV_2D	LEAKY_RELU	RELU	RELU6
FULLY_CONNECTED	LOCAL_RESPONSE_NORMALIZATION	PAD	PADV2	PRELU	QUANTIZE	DEQUANTIZE	REDUCE_MAX	REDUCE_MIN
REDUCE_PROD	SUM	RESHAPE	SQUEEZE	RESIZE_NEAREST_NEIGHBOR	RESIZE_BILINEAR	SLICE	LOG_SOFTMAX	SOFTMAX
POW	MUL	MINIMUM	FLOOR_MOD	FLOOR_DIV	ADD	SPLIT	STRIDED_SLICE	TRANSPOSE

life.augmented

ONNX

- In *ONNX* a subset of operators from Opset 7, 8, 9 and 10 of ONNX 1.6 is supported.
- Model may be loaded from a single file with model and weights (.onnx).

ONNX operators supported by cube.AI								
Add	AveragePool	BatchNormalization	Concat	Constant	Conv	ConvTranspose	Div	Elu
Flatten	Gemm	Hardmax	HardSigmoid	GlobalAveragePool	GlobalMaxPool	InstanceNormalization	LeakyReLU	LogSoftmax
LpNormalization	LRN	MatMul	MaxPool	Max	Mul	Pad	PRelu	Reshape
ReduceMax	Resize	Selu	Slice	Squeeze	Softmax	Tile	ThresholdedRelu	Transpose

life.augmented

Introduction to FP-AI-VISION1

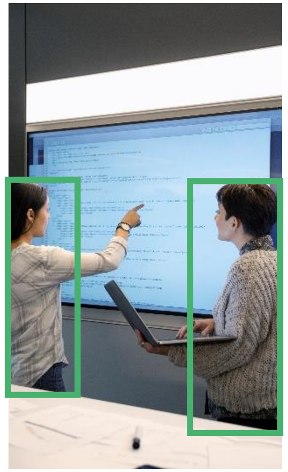
life.augmented

FP-AI-VISION1

Give vision to your STM32 product for new features and add-on services

Food classification

Person presence detection



People counting

FP-AI-VISION1 v1.0 → FP-AI-VISION1 v2.0 → FP-AI-VISION1 v3.0

AI applications

Middleware
- STM32_AI_Runtime
- STM32_Image
- STM32_AI_Utilities
- STM32_USB
- STM32_Fs
- FatFs

Drivers
- BSP
- HAL

Hardware component
- CAMERA
- STM32
- LCD

Development boards
- B-CAMS-OMV
- STM32H747I-DISCO

life.augmented

FP-AI-VISION1

Person presence detection

Replace PIR with Ultra-low power and reliable person detection

- Visual wake word for smart homes or city security cameras
- Multiple models suitable for **ultra-low power STM32L4R to high performance STM32H7 MCUs,** depending on required performance and cost
- **Reduce false alarms** due to object movement detection

STM32 Cube.AI

Visual Wake Word model on STM32L4R

Model input resolution	96x96 RGB pixels
Model complexity	69k MACC
Inference time	274 ms @ 120 MHz
Max rate	3.6 FPS
Flash	214 KB
RAM	46 KB
MCU power consumption (full FPS)	35 mA
MCU power consumption (SMPS)	22 mA

life.augmented

FP-AI-VISION1

Image classification

Apply image classification to your own use case

- CNN can classify 18 types of food (224x224 RGB images), but you can **retrain with your own dataset** for defect detection, material classification, conveyor belt sorting and more!
- Multiple examples of camera input resolution and quantization for accuracy/footprint tradeoff optimization
- Different memory mappings to optimize and test impact on performances

STM32 Cube.AI

Food classification demo on STM32H7*

Model input resolution	QVGA
Model complexity	69k MACC
Inference time	145 ms @ 400 MHz
Max rate	5.4 FPS
Flash	160 KB
RAM	240 KB
MCU power consumption (SMPS)	80 mA

*memory optimized model
Other are available in FP-AI-VISION1

life.augmented

FP-AI-VISION1

People counting

Monitor building usage with cost and power efficient solution

- Detect multiple people to enable your system to count **accurately**
- Add intelligence to your **smart building** : monitor factory, meeting room or showroom people flows
- **Monitor physical distances** between multiple people

STM32 Cube.AI

Advanced models on STM32H7

Model input resolution	240x240 RGB pixels
Model complexity	96M MACC
Inference time	371 ms @ 400 MHz
Max rate	2.7 FPS
Flash	230 KB
RAM	233 KB
MCU power consumption (SMPS)	80 mA

life.augmented

FP-AI-VISION1

- FP-AI-VISION1 is a function pack (FP) demonstrating the capability of STM32H7 Series microcontrollers to execute a Convolutional Neural Network (CNN) efficiently in relation to computer vision tasks. FP-AI-VISION1 contains everything needed to build a CNN-based computer vision application on STM32H7 microcontrollers.
- FP-AI-VISION1 can be downloaded from the link below
- https://www.st.com/content/st_com/en/search.html#q=fp-ai-vision1-t=tools-page=1

Name	Date modified	Type	Size
_htmresc	2021/2/18 4:38	File folder	
Documentation	2021/2/18 5:39	File folder	
Drivers	2021/2/18 4:38	File folder	
Middlewares	2021/2/18 4:38	File folder	
Projects	2021/2/18 4:38	File folder	
Utilities	2021/2/18 4:38	File folder	
License.md	2021/2/5 3:31	MD File	2 KB
Release_Notes.html	2021/2/18 4:55	Chrome HTML Docu...	46 KB

life.augmented

FP-AI-VISION1 main feature

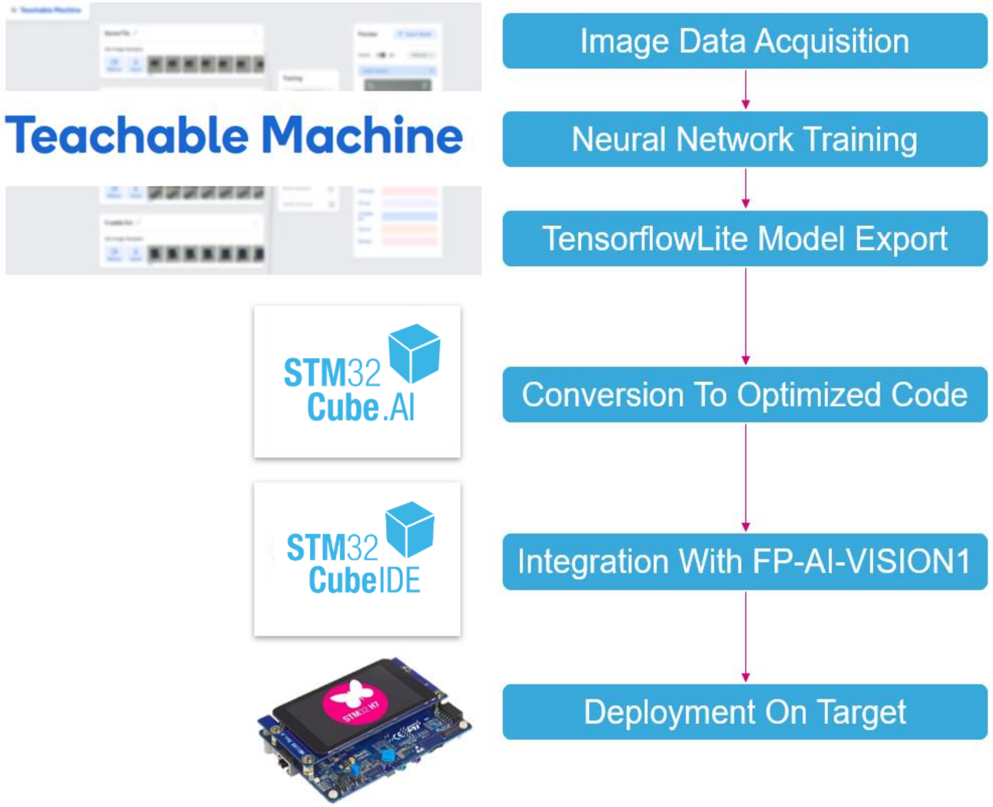
- Runs on the STM32H747I-DISCO board connected with the STM32F4DIS-CAM camera daughterboard
- Includes three image classification application examples based on CNN:
  - One food recognition application operating on color (RGB 24 bits) frame images
  - One person presence detection application operating on color (RGB 24 bits) frame images
  - One person presence detection application operating on grayscale (8 bits) frame images
- Includes complete application firmware for camera capture, frame image preprocessing, inference execution
- and output post-processing
  - Includes examples of integration of both floating-point and 8-bit quantized C models
  - Supports several configurations for data memory placement in order to meet application requirements

STM32H747I-DISCO

B-CAMS-OMV

How to use Teachable Machine to create an image classification application on STM32

- The first part shows how to use the Teachable Machine to train and export a deep learning model, then STM32Cube.AI is used to convert this model into optimized C code for STM32 MCUs. The last part explains how to integrate this new model into the FP-AI-VISION1 to run live inference on an STM32 board with a camera. The whole process is described below:



life.augmented

Prerequisites

- Before the experiment, we need some preparation of software and hardware.

- STM32Cube IDE
- X-Cube-AI version 7.1.0
- FP-AI-VISION1 version 3.1.0
- STM32CubeProgrammer

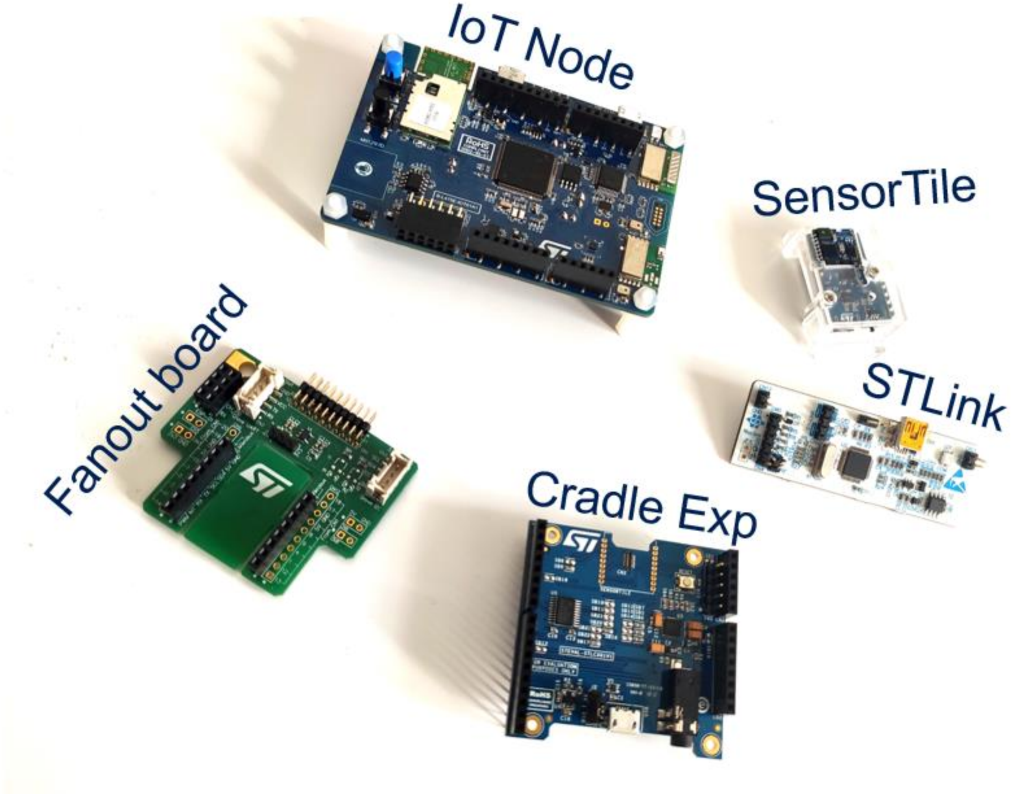
- STM32H747I-DISCO Board
- B-CAMS-OMV Flexible Camera Adapter board
- A Micro-USB to USB cable

https://wiki.stmicroelectronics.cn/stm32mcu/index.php?title=AI:How_to_use_Teachable_Machine_to_create_an_image_classification_application_on_STM32&icmp=tt19900_gl_pron_feb2021

life.augmented

Training a model using Teachable Machine

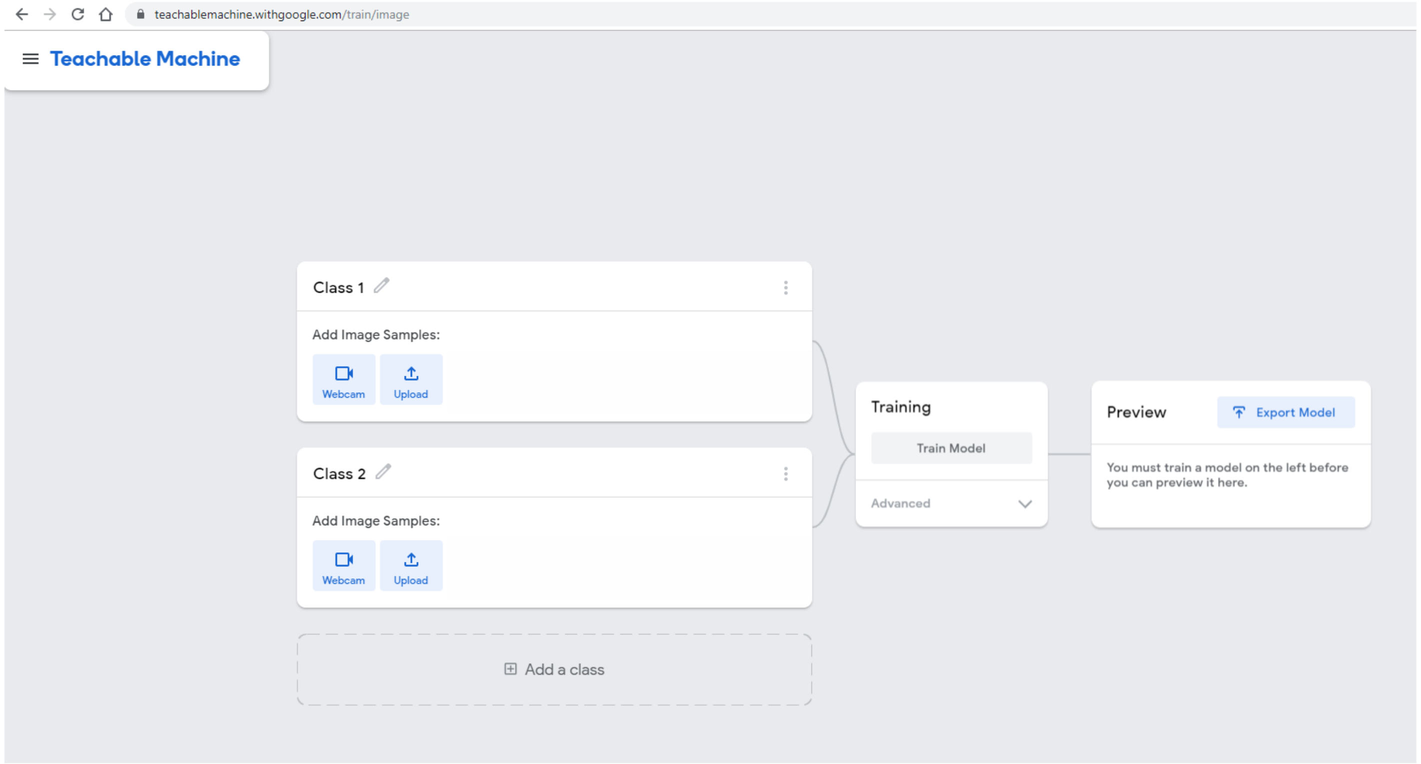
- We first need to choose something to classify. You can choose whatever object you want to classify it: fruits, pasta, animals, people, etc...
- In this example, we will classify ST boards and modules. The chosen boards are shown in the figure below:



If you are interested in replicating this example you can purchase the ST eval boards mentioned

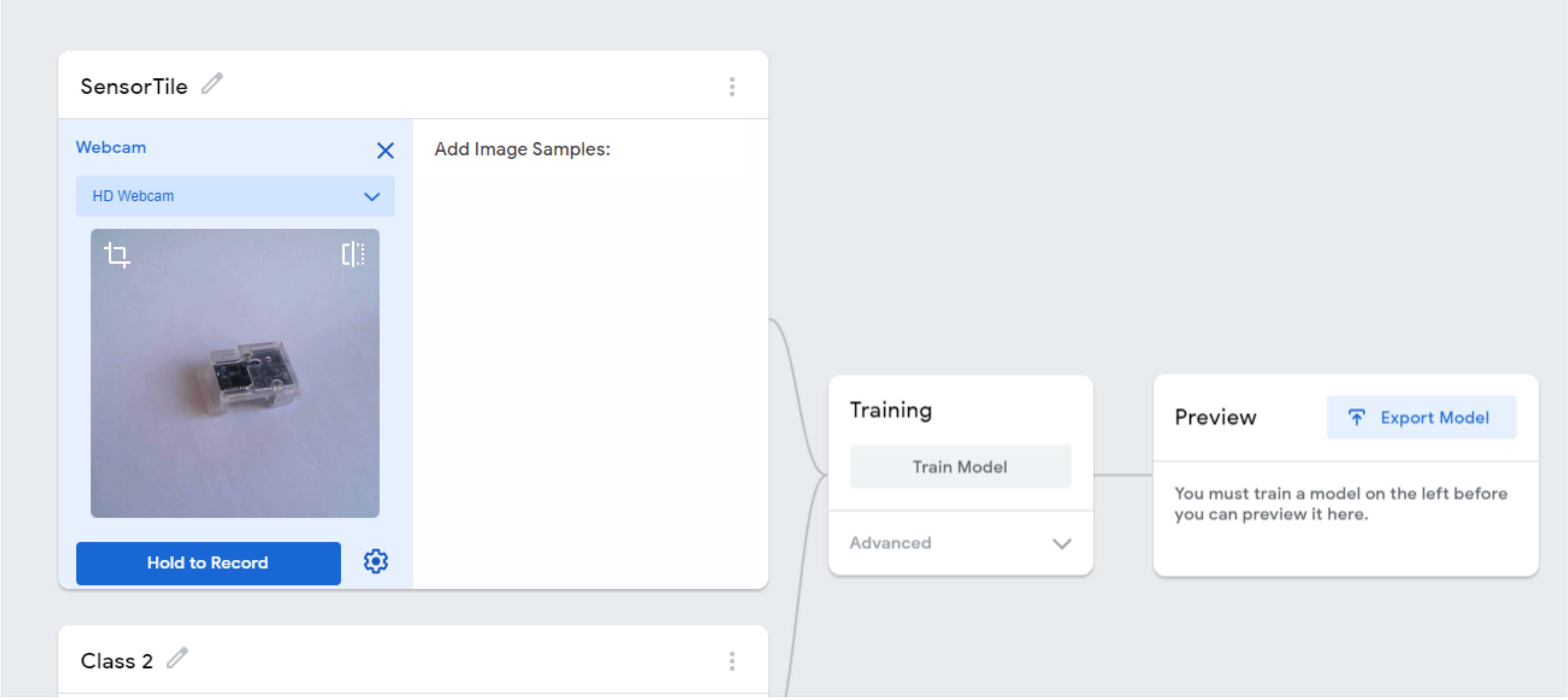
Training a model using Teachable Machine

- Let's get started. Open https://teachablemachine.withgoogle.com/, preferably from Chrome browser.
- Click **Get started**, then select **Image Project**. You will be presented with the following interface.



Training a model using Teachable Machine

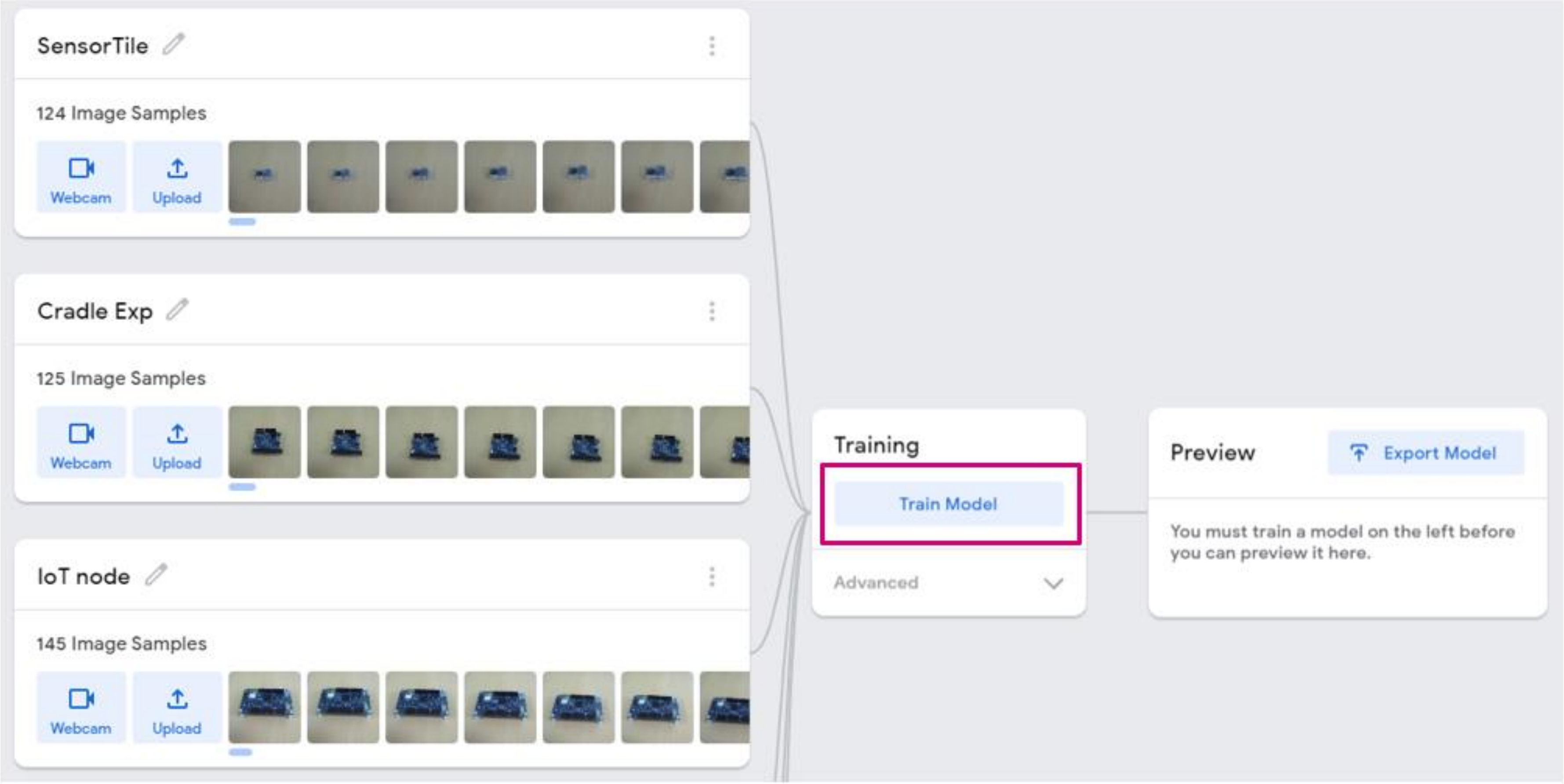
- For each category you want to classify, edit the class name by clicking the pencil icon. In this example, we choose to start with SensorTile.
- To add images with your webcam, click the webcam icon and record some images. If you have image files on your computer, click upload and select the directory containing your images.



life.augmented

Training a model using Teachable Machine

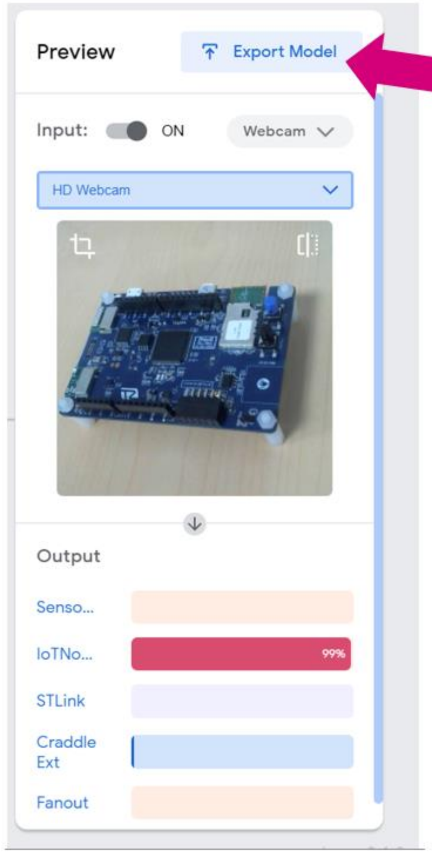
- Now that we have a good amount of data, we are going to train a deep learning model for classifying these different objects. To do this, click the Train Model button as shown below:

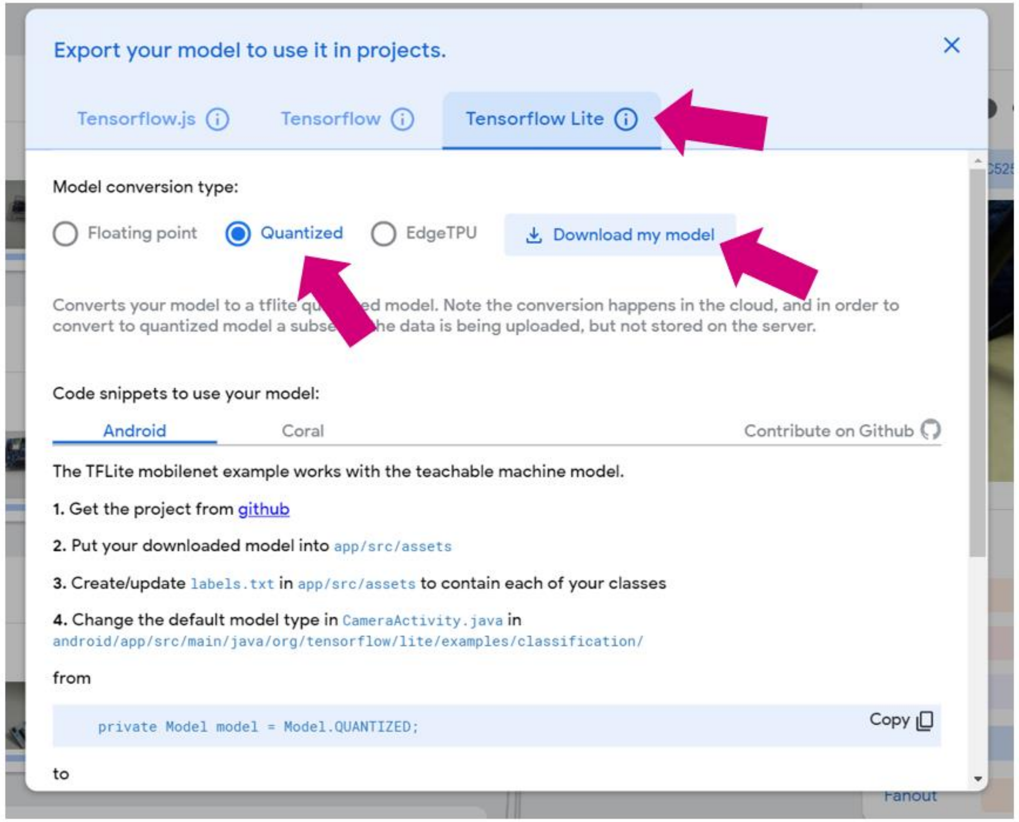


life.augmented

Training a model using Teachable Machine

- If you are happy with your model, it is time to export it. To do so, click the **Export Model** button. In the pop-up window, select **Tensorflow Lite**, check **Quantized** and click **Download my model**.





Porting to a target board

- use the stm32ai command line tool to convert the TensorflowLite model to optimized C code for STM32.

```
stm32ai generate -m model.tflite -v 2
```

- The expected output is:

```
Neural Network Tools for STM32 v1.3.0 (AI tools v5.1.0)
Running "generate" cmd...
-- Importing model
 model files : /path/to/workspace/model.tflite
 model type  : tflite (tflite)
-- Importing model - done (elapsed time 0.531s)
-- Rendering model
-- Rendering model - done (elapsed time 0.184s)
-- Generating C-code
Creating /path/to/workspace/stm32ai_output/network.c
Creating /path/to/workspace/stm32ai_output/network_data.c
Creating /path/to/workspace/stm32ai_output/network.h
Creating /path/to/workspace/stm32ai_output/network_data.h
-- Generating C-code - done (elapsed time 0.782s)

Creating report file /path/to/workspace/stm32ai_output/network_generate_report.txt
```

Porting to a target board

- use the stm32ai command line tool to convert the TensorflowLite model to optimized C code for STM32.

```
stm32ai generate -m model.tflite -v 2
```

- The expected output is:

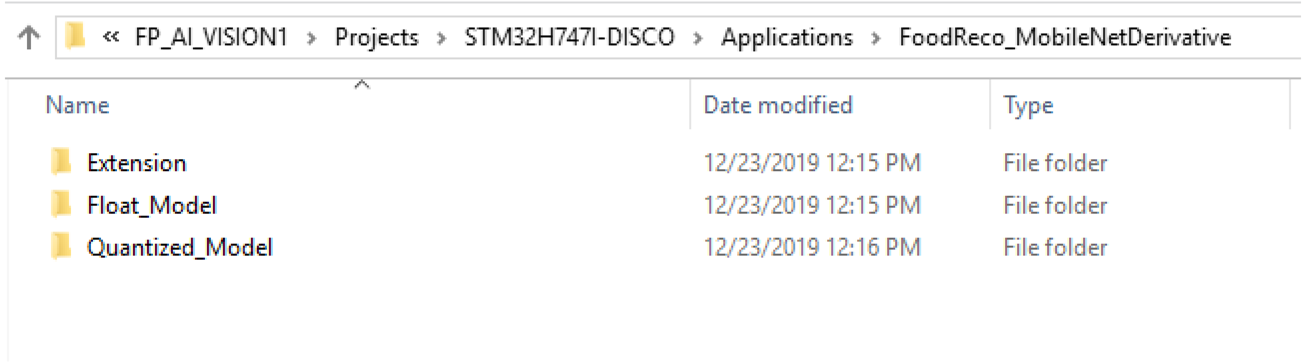
```
Neural Network Tools for STM32 v1.3.0 (AI tools v5.1.0)
Running "generate" cmd...
-- Importing model
 model files : /path/to/workspace/model.tflite
 model type  : tflite (tflite)
-- Importing model - done (elapsed time 0.531s)
-- Rendering model
-- Rendering model - done (elapsed time 0.184s)
-- Generating C-code
Creating /path/to/workspace/stm32ai_output/network.c
Creating /path/to/workspace/stm32ai_output/network_data.c
Creating /path/to/workspace/stm32ai_output/network.h
Creating /path/to/workspace/stm32ai_output/network_data.h
-- Generating C-code - done (elapsed time 0.782s)

Creating report file /path/to/workspace/stm32ai_output/network_generate_report.txt
```

This command generates four files under workspace/stm32ai_ouptut/:

life.augmented

Integration with FP-AI-VISION1

- the FP-AI-VISION1 function pack provides a software example for a food classification application
- The main objective of this section is to replace the network and network_data files in FP-AI-VISION1 by the newly generated files and make a few adjustments to the code.
- If we take a look inside the function pack, we'll start from the FoodReco_MobileNetDerivative application we can see two configurations for the model data type, as shown below.
- Delete the following files and replace them with the ones from workspace/stm32ai_output:



In Src:
network.c
network_data.c

In Inc:
network.h
network_data.h

Updating the labels and display

- From STM32CubeIDE, open fp_vision_app.c. Go to line 125 where the output_labels is defined and update this variable with our label names:

```
// fp_vision_app.c line 125
const char* output_labels[AI_NET_OUTPUT_SIZE] = {
    "SensorTile", "IoTNode", "STLink", "Craddle Ext", "Fanout", "Background"};
```

- While we're here, we'll update the display mode that it shows camera image instead of food logos. Go around line 200 and update the App_Output_Display function. At the top of the function, the display_mode variable should be set to 1.

```
static void App_Output_Display(AppContext_TypeDef *App_Context_Ptr)
{
  static uint32_t occurrence_number = NN_OUTPUT_DISPLAY_REFRESH_RATE;
  static uint32_t display_mode = 1; // Was 0
```

life.augmented

Cropping the image

- Teachable Machine crops the webcam image to fit the model input size. In FP-AI-VISION1, the image is resized to the model input size, hence losing the aspect ratio. We will change this default behavior and implement a crop of the camera image.

- In order to have square images and avoid image deformation we are going to crop the camera image using the DCMI. The goal of this step is to go from the 640x480 resolution to a 480x480 resolution.

- First, edit fp_vision_camera.h line 60 to update the CAMERA_WIDTH define to 480 pixels:

```
//fp_vision_camera.h line 57
#if CAMERA_CAPTURE_RES == VGA_640_480_RES
#define CAMERA_RESOLUTION CAMERA_R640x480
#define CAM_RES_WIDTH 480 // Was 640
#define CAM_RES_HEIGHT 480
```

Cropping the image

- Then, edit fp_vision_camera.c located in Application/.
- Modify the CAMERA_Init function (line 58) to configure DCMI cropping (update the function with the highlighted code bellow) :

```
  /* Set camera mirror / flip configuration */
  CAMERA_Set_MirrorFlip(Camera_Context_Ptr, Camera_Context_Ptr->mirror_flip);

  /* If image was flipped, set the option here (no flip by default) */
  /* uncomment the line below */
  /* CAMERA_Set_MirrorFlip(Camera_Context_Ptr, CAMERA_MIRRORFLIP_FLIP); */

  HAL_Delay(100);

 /* If image was flipped, force the option
  /* Center-crop the 640x480 frame to 480x480 */
  const uint32_t x0 = (640 - 480) / 2;
  const uint32_t y0 = 0;

  /* Note: 1 px every 2 DCMI_PXCLK (8-bit interface in RGB565) */
  HAL_DCMI_ConfigCrop(&hcamera_dcmi,
                      x0 * 2,
                      y0,
                      CAM_RES_WIDTH * 2 - 1,
                      CAM_RES_HEIGHT - 1);

  HAL_DCMI_EnableCrop(&hcamera_dcmi);
```

life.augmented

Normalization

- The neural network input needs to be normalized accordingly to the training phase. This is achieved by updating the value of both the nn_input_norm_scale and nn_input_norm_zp variables during initialization. The nn_input_norm_scale and nn_input_norm_zp variables affect the pixel format adaptation stage. The scale, zero point values should be set {127.5, 127} if the NN model was trained using input data normalized in the range [-1, 1]. They should be set to {255, 0} if the NN model was trained using input data normalized in the range [0, 1]. The food recognition model was trained with input data normalized in the range [0, 1] whereas the Teachable Model was trained in the range of [-1, 1].

- Edit the file fp_vision_app.c and modify the App_Context_Init function (line 328) to update the scale and zero-point values (update the function with the highlighted code bellow) :

```
/*{scale,zero-point} set to {127.5, 127} since NN model was trained using input data normalized in the range [-1, 1]*/
App_Context_Ptr->Ai_ContextPtr->nn_input_norm_scale=127.5f; //was 255.0f
App_Context_Ptr->Ai_ContextPtr->nn_input_norm_zp=127; //was 0
```

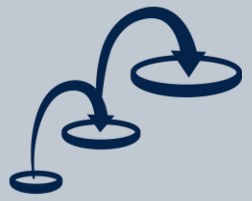
Testing the model

- Compiling the project and then Connect the STM32H747I-DISCO to your PC via a Micro-USB to USB cable. Open STM32CubeProgrammer and connect to ST-LINK. Then flash the board with the hex file.
- Connect the camera to the STM32H747I-DISCO board using a flex cable. To have the image in the upright position, the camera must be placed with the flex cable facing up as shown in the figure below. Once the camera is connected, power on the board and press the reset button. After the "Welcome Screen", you will see the camera preview and output prediction of the model on the LCD Screen.

life.augmented

We provide everything to kick off your project

Design documentation



Getting started

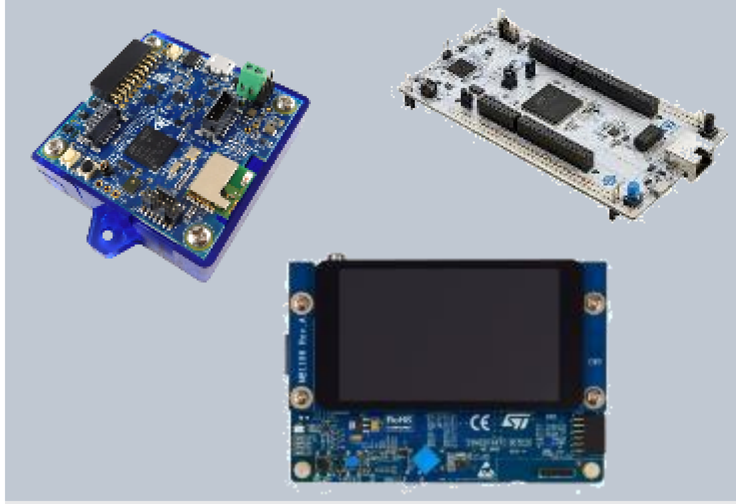
Be guided step-by-step to learn STM32 ecosystem

Developer zone

Get started on application development and project sharing

- **Wiki by ST** is a great forum to learn and start developing AI on STM32!
- Videos of application examples
- Massive Open Online Course (MOOC)

Hardware and software tools



- Evaluation platforms for STM32 MCU/MPU
- Extra sensor boards
- Full software suite

Support & Updates

- **ST Community**: STM32 ML & AI group
- Distributor certified FAE
- Support center
- Newsletter

Our technology starts with You

Find out more at www.st.com/stm32ai

life.augmented