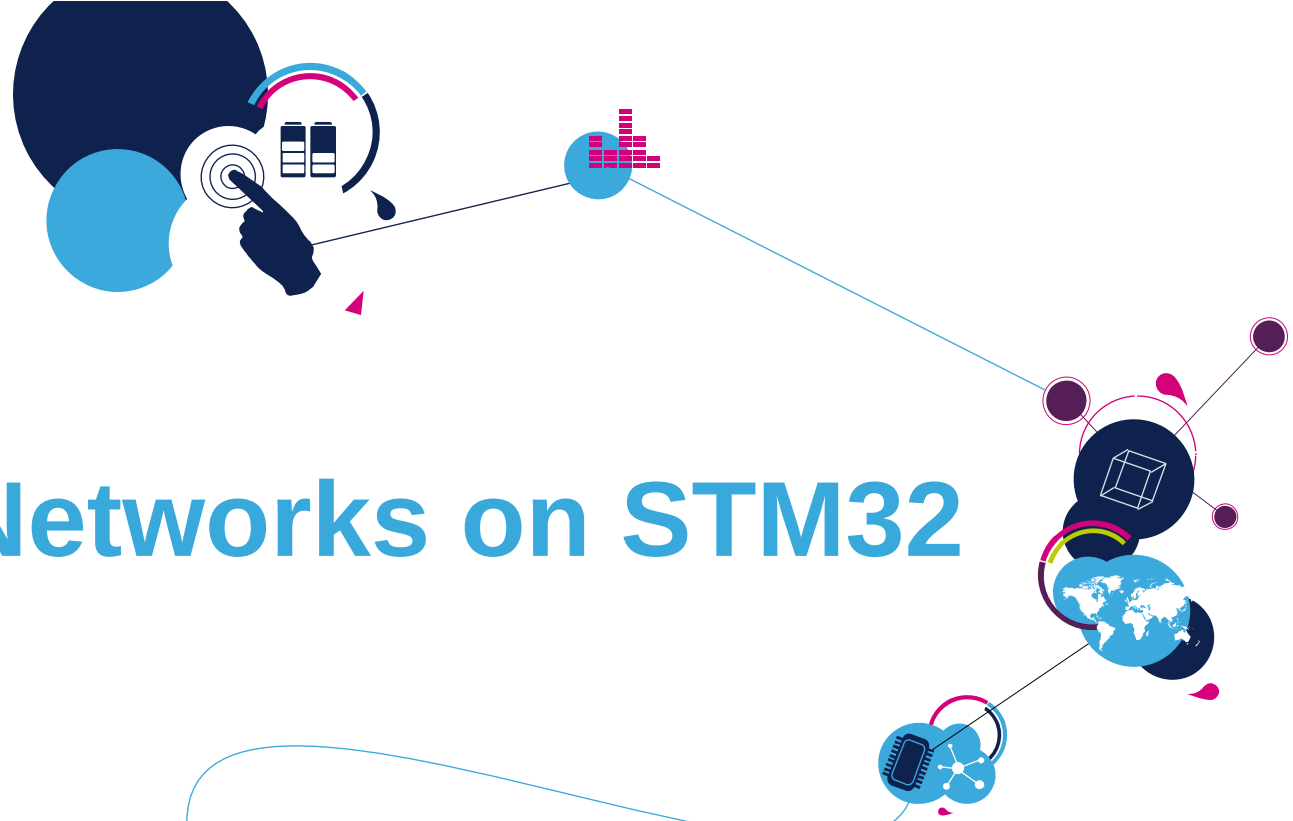


Implementing Neural Networks on STM32

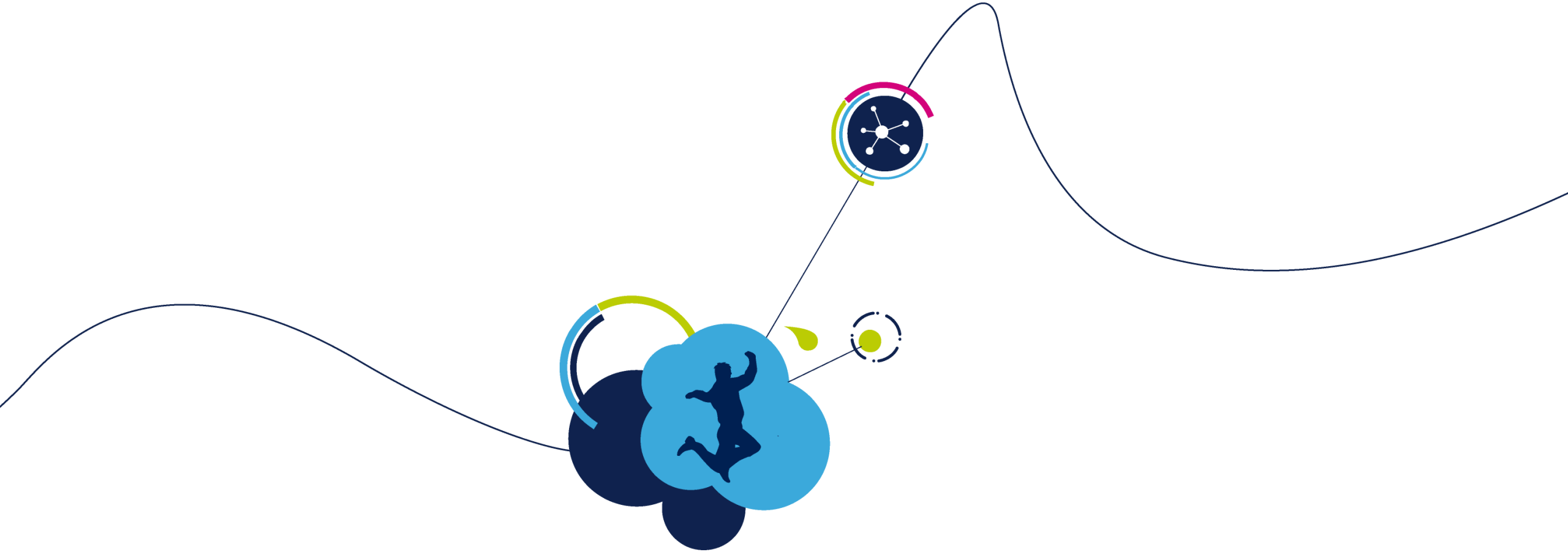
life.augmented

Target

- WE will provide you an introduction to NN approach implemented by ST
  - Positioning of NN within Artificial Intelligence systems
  - Artificial neuron basic model
  - An example of the bridge between biological and Artificial NN
  - NN implementation chain proposed by ST
  - Collection and labeling of NN training data set
  - Cube.AI hands-on
  - Building and training of NN (trainer presentation)
  - FP-AI-SENSING1 software pack explanation and modification
- YOU will be in a position to understand and implement it

Agenda

- **09:00** Image Recognition live demo
- **09:20** Introduction to AI and NN
- **10:10** Lab1: Out of the box
- ***10:40 Break***
- **11:10** The Five steps development behind NN
- **11:40** Lab2: How to collect & label learning dataset
- ***12:10 Lunch***
- **13:10** Live Demo : NN model creation & learning
- **13:55** Time for Partners
- **14:55** STM32CubeMx short introduction
- ***15:15 Break***
- **15:45** Lab3: STM32CubeMX & X-CUBE-AI
- **16:30** Lab4: How to update modified NN model within your application
- **16:50** – 17:00 Conclusion and wrap-up



Let’s start with image recognition…

life.augmented

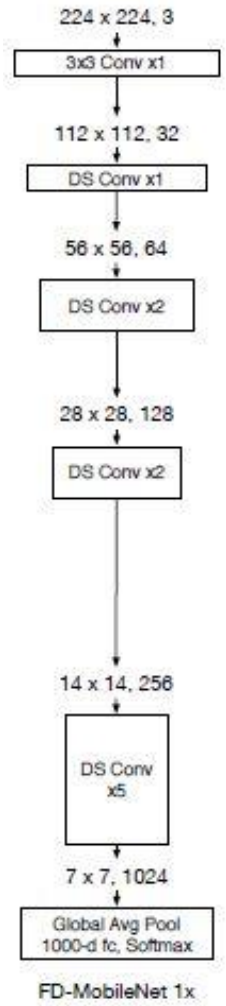
Fast Downsampling MobileNet Food Recognition on STM32H747 Discovery board

Neural Network

- FD-MobileNet topology from public paper
- Mixed dataset Food-101, FoodNet, ST

Implementation

- Exploits Camera in continuous mode 5.5 fps or one shot
- Floating Point or mixed model3 Float/Fix Point
- 18 food classes
- Pre-processing: rescaling from 640x480 to 224x224 RGB 8 bit image



FD-MobileNet 1x

STM32 Cube.AI

STM32 Cube.AI NN

- Memory footprint: 205 KB RAM, 191 KB Flash

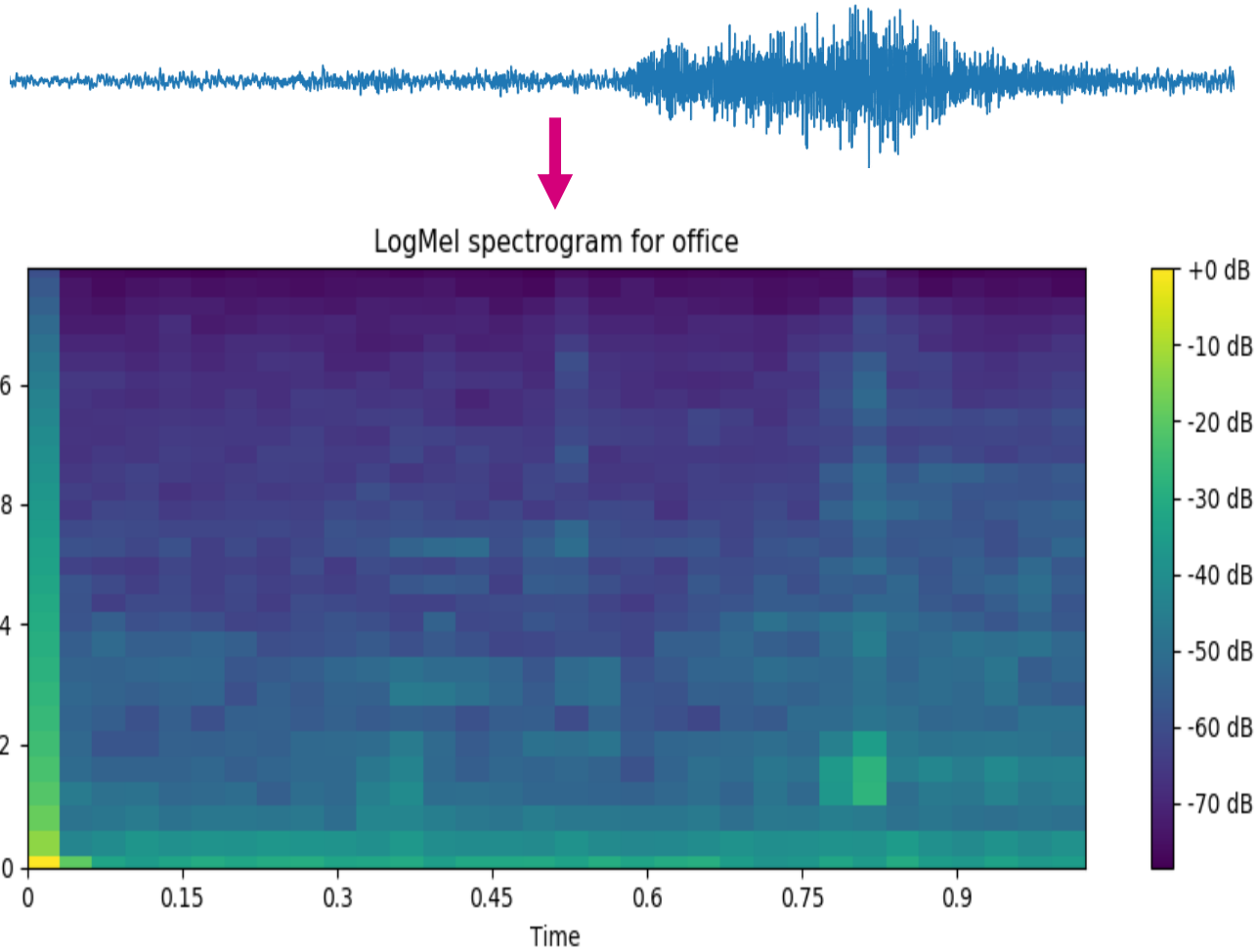
STM32 H7

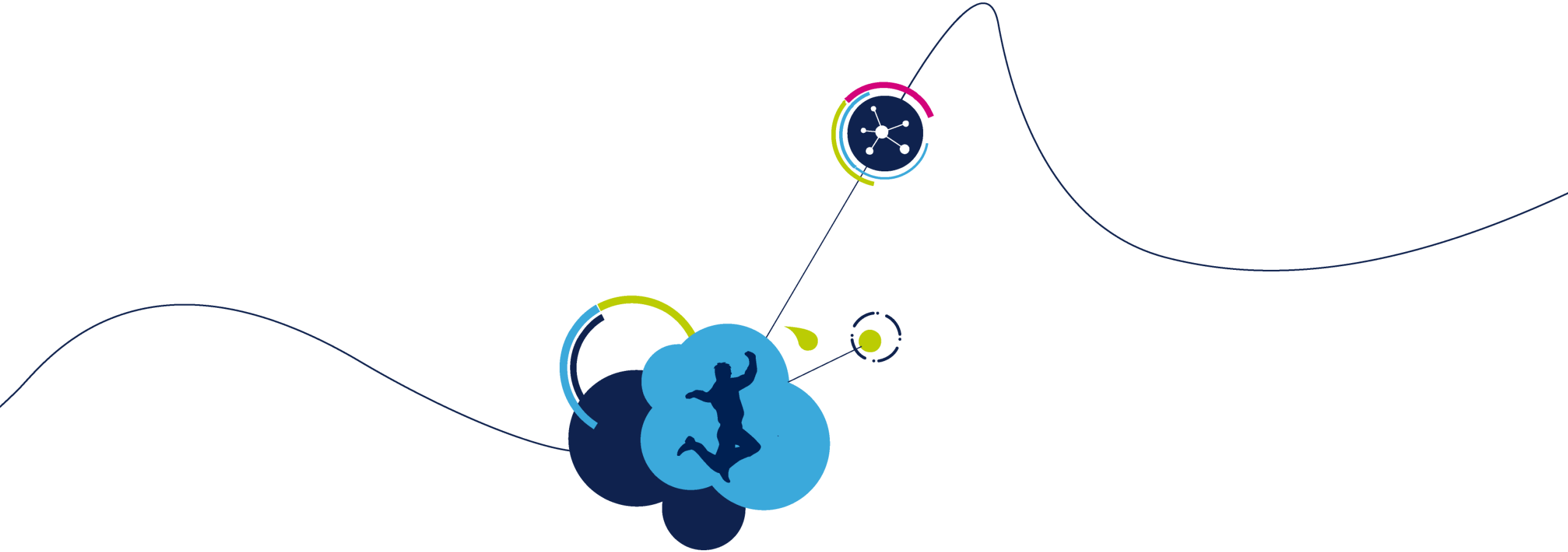
Performance on STM32H747

- 1 inference per image
- STM32H747 400 MHz Cortex-M7F
- Mix model Fix/Floating Point
  - 60 MHz / 150 ms per inference
  - Accuracy: 78.8% (vs 77.7% in float)

life.augmented

Image recognition by ANN





Introduction to Artificial Intelligence and Neural Networks

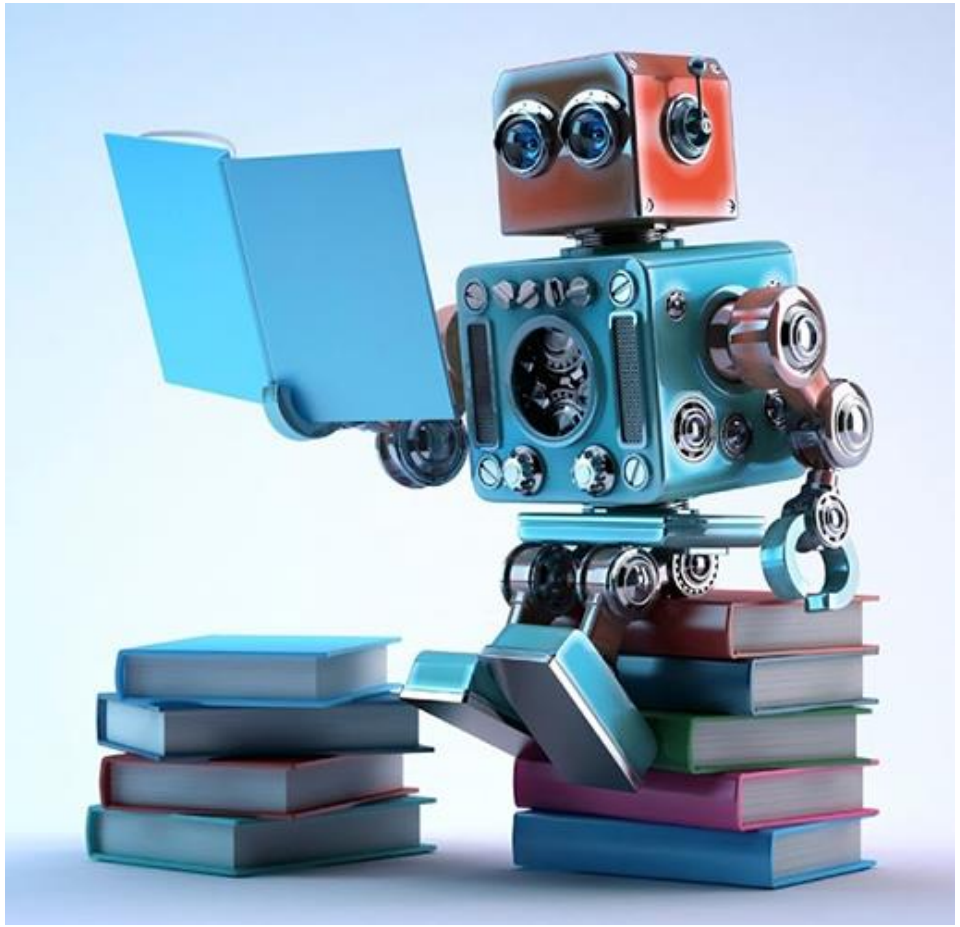
life.augmented

Artificial Intelligence (AI)

- AI is a superset of all the studies where machines mimic cognitive capabilities like humans. For example:
  - Interaction with the environment
  - Knowledge representation
  - Perception
  - Learning
  - Computer vision
  - Speech recognition
  - Problem solving
- Uses concepts from
  - Computer science
  - Statistics
  - Mathematics

Machine Learning (ML)

- ML is a sub-branch of AI;
- The field of computer science that gives computers the ability to learn without being explicitly programmed.
- Consists of algorithms that can learn and make predictions on data. Such algorithms are:
  - trained on previous examples to build a model
  - usually employed where traditional programming is infeasible;
  - should work for new cases if trained properly

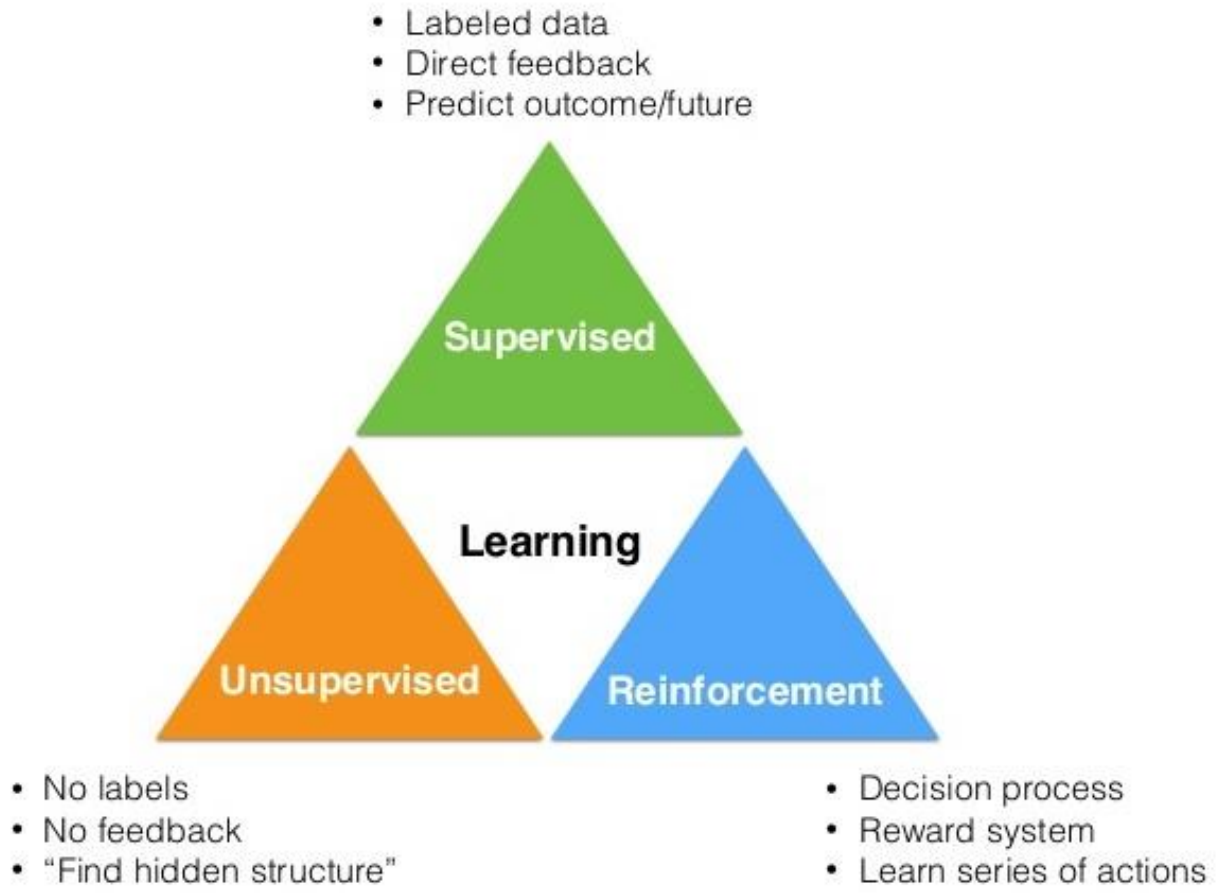


Machine Learning (ML)

ML Application Areas

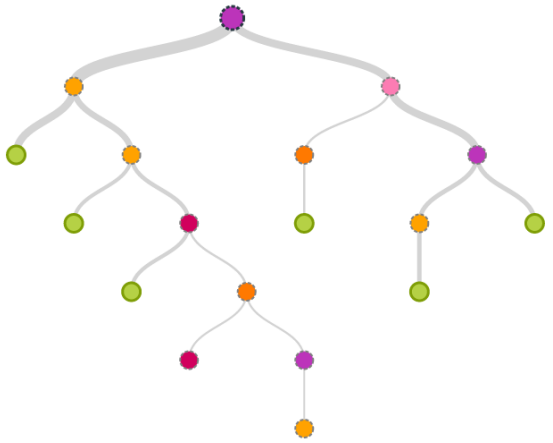
- Pattern recognition
  - Object in real scenes
  - Facial identities/expressions
  - Spoken words
- Anomaly recognition
  - Fraudulent credit card transactions
  - Unusual patterns of sensor reading in nuclear plants
- Predicting future trends
  - Future stock prices
  - Targeted advertisements

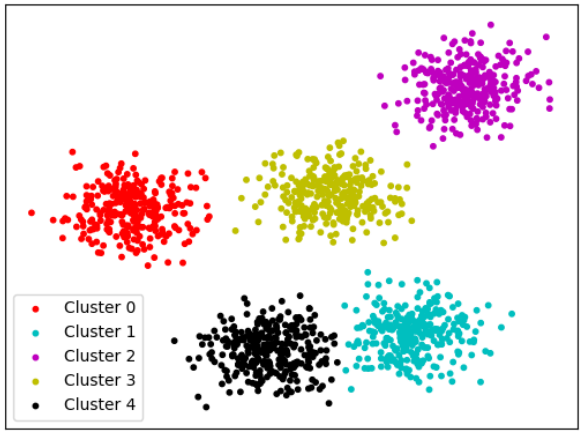
ML Types

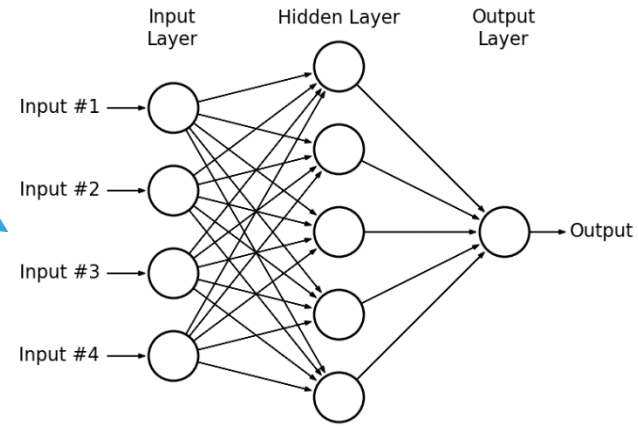


Some approaches to machine learning

- Decision tree learning
- Clustering
- Rule based learning
- Inductive logic programming → Prolog
- Deep learning → NN



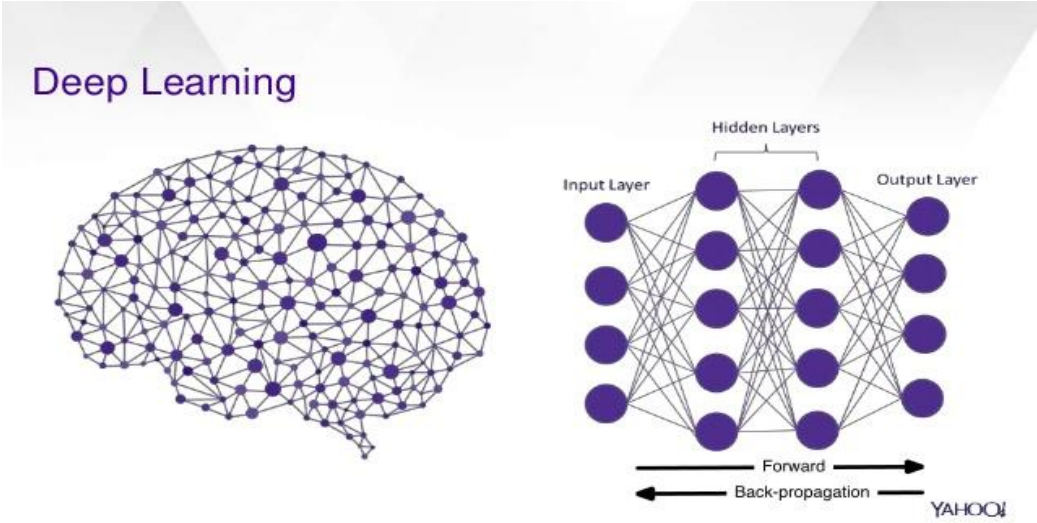




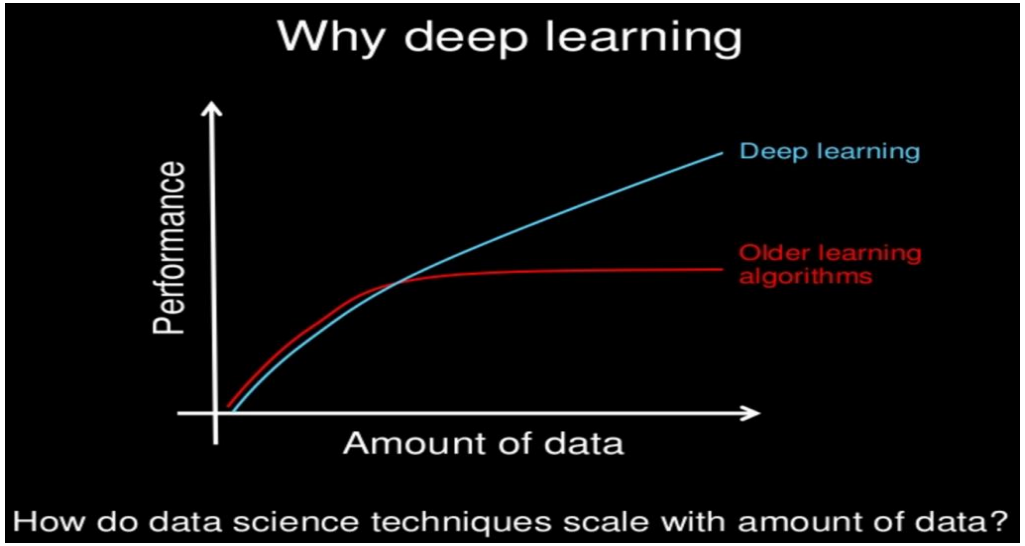
life.augmented

Deep Learning (DL)

- Deep learning is ML using neural networks.
  - Inspired by biological neural networks
  - Deep because of having many intermediate learning steps.
  - Lots and lots of data is required



Advantages	Disadvantages
Autonomous Learning of data patterns & relationships	Large Datasets
High Accuracy	High Computational Requirements
Easy Improvement & Fine Tuning	Weak theoretical explanation
Adaptive Solutions	Black box (for most people)

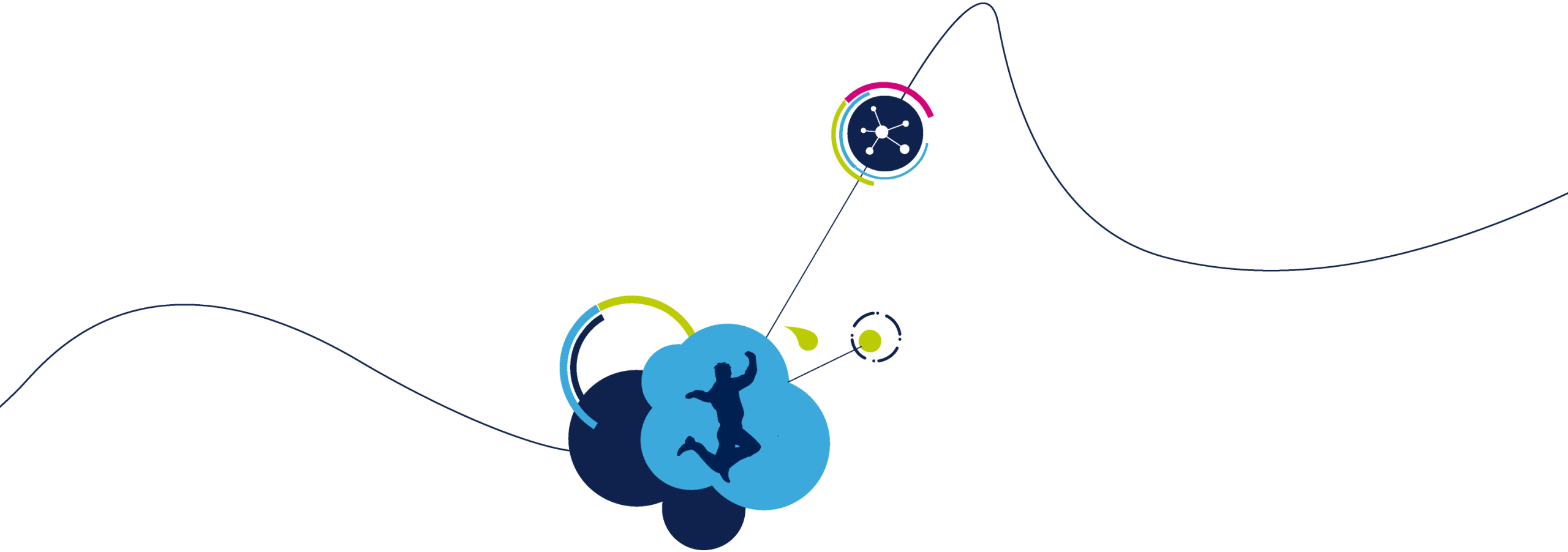


Why Deep Learning is so important

- Convolutional Deep Neural Networks outperform previous methods on a number of tasks:

Problem	Dataset	Best Accuracy w/o CNN	Best Accuracy with CNN	Diff
Object classification	ILSVRC	**73.8%**	**95.1%**	**+21.3%**
Scene classification	SUN	**37.5%**	**56%**	**+18.5%**
Object detection	VOC 2007	**34.3%**	**60.9%**	**+26.6%**
Fine-grained class	200Birds	**61.8%**	**75.7%**	**+13.9%**
Attribute detection	H3D	**69.1%**	**74.6%**	+5.5%
Face recognition	LFW	**96.3%**	**99.77%**	**+3.47%**
Instance retrieval	UKB	**89.3%** (CDVS: 85.7%)	**96.3%**	+7.0%

May 2015

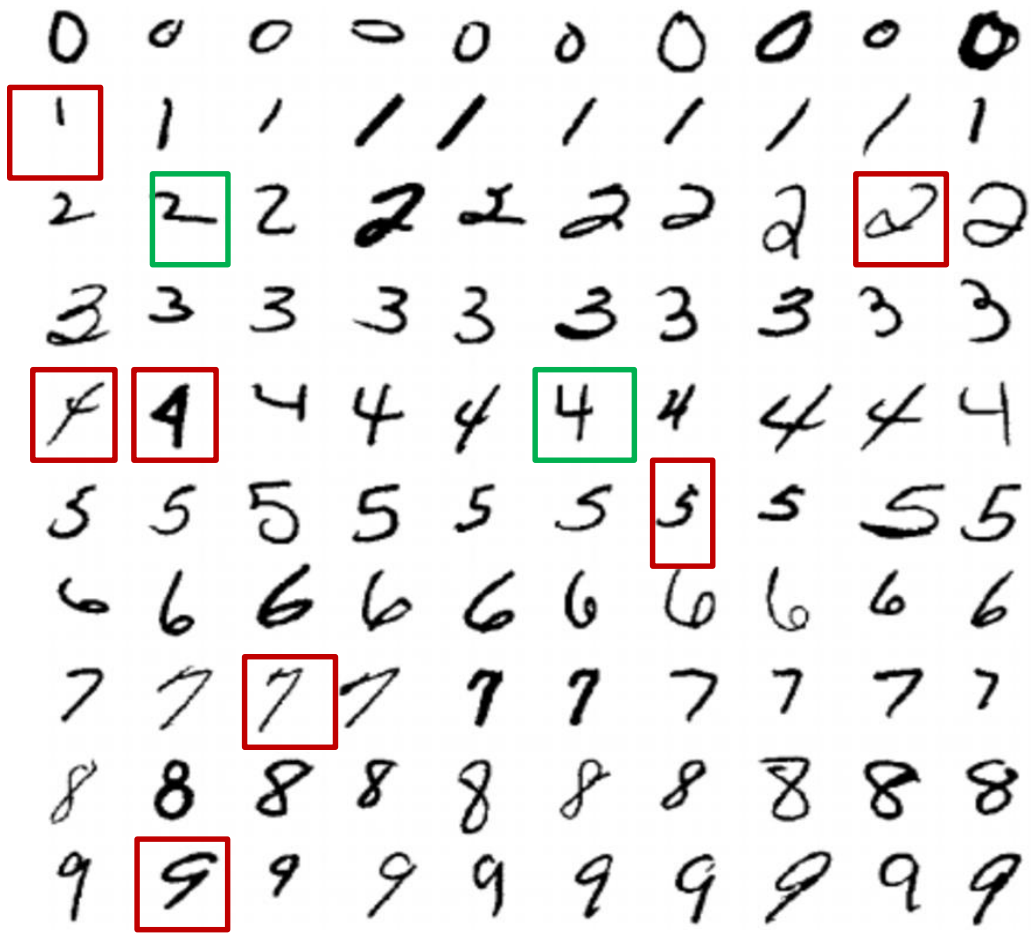


Machine Learning: the reason why?

life.augmented

Why do we need ML ?

- An example of complex problem
  - How to recognize the hand written digits?
  - Very difficult to define the rules!
  - What makes all these numbers to be identifiable?
  - Is there a pattern?
  - What is it that makes a 2 to be identified as a 2?



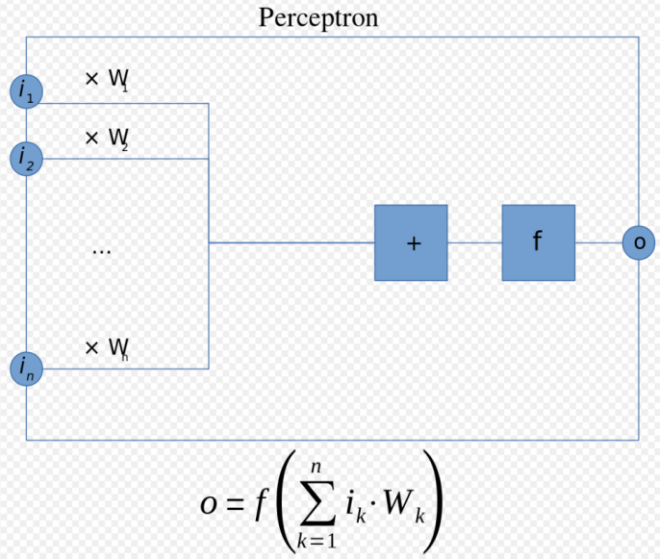
Some examples from MNIST database

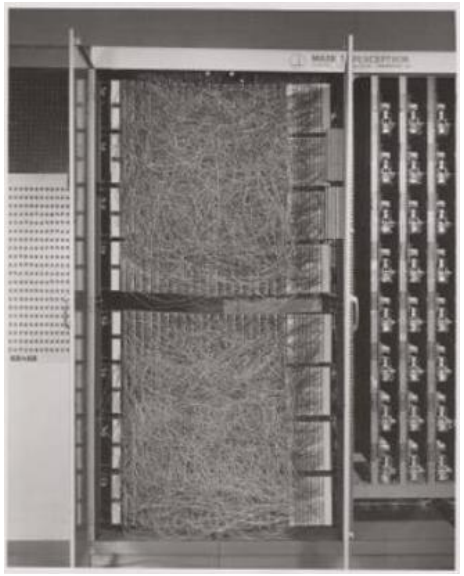
(Mixed standard institute for standard and technology)

Is it “cutting edge” technology ?

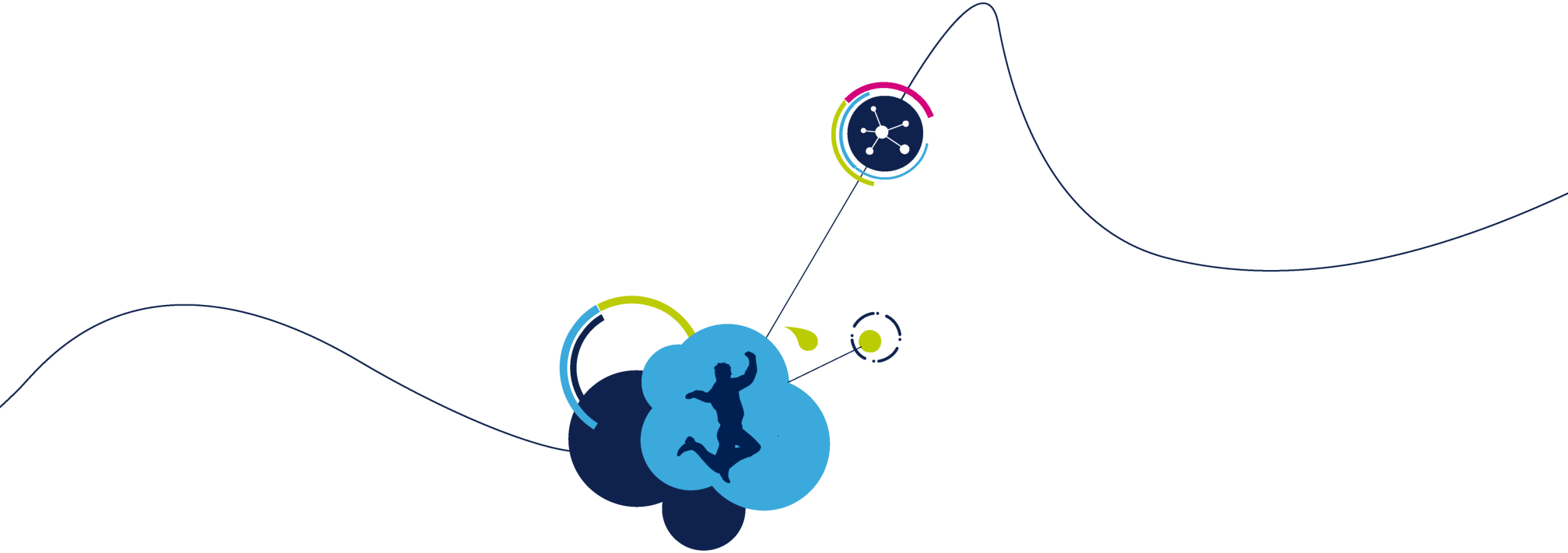
The “perceptron” image recognition algorithm is based on simple NN.
It has been invented by Frank Rosenblatt in **1957**.

Today, the perceptron concept is a part of **M**ulti **L**ayer **P**erceptron NN.





Source: https://en.wikipedia.org/wiki/Perceptron

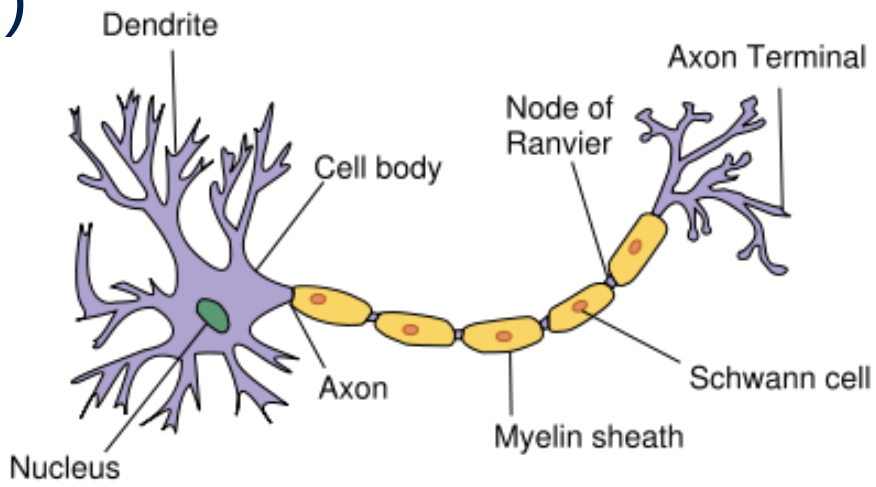


Introduction to Neural Networks

life.augmented

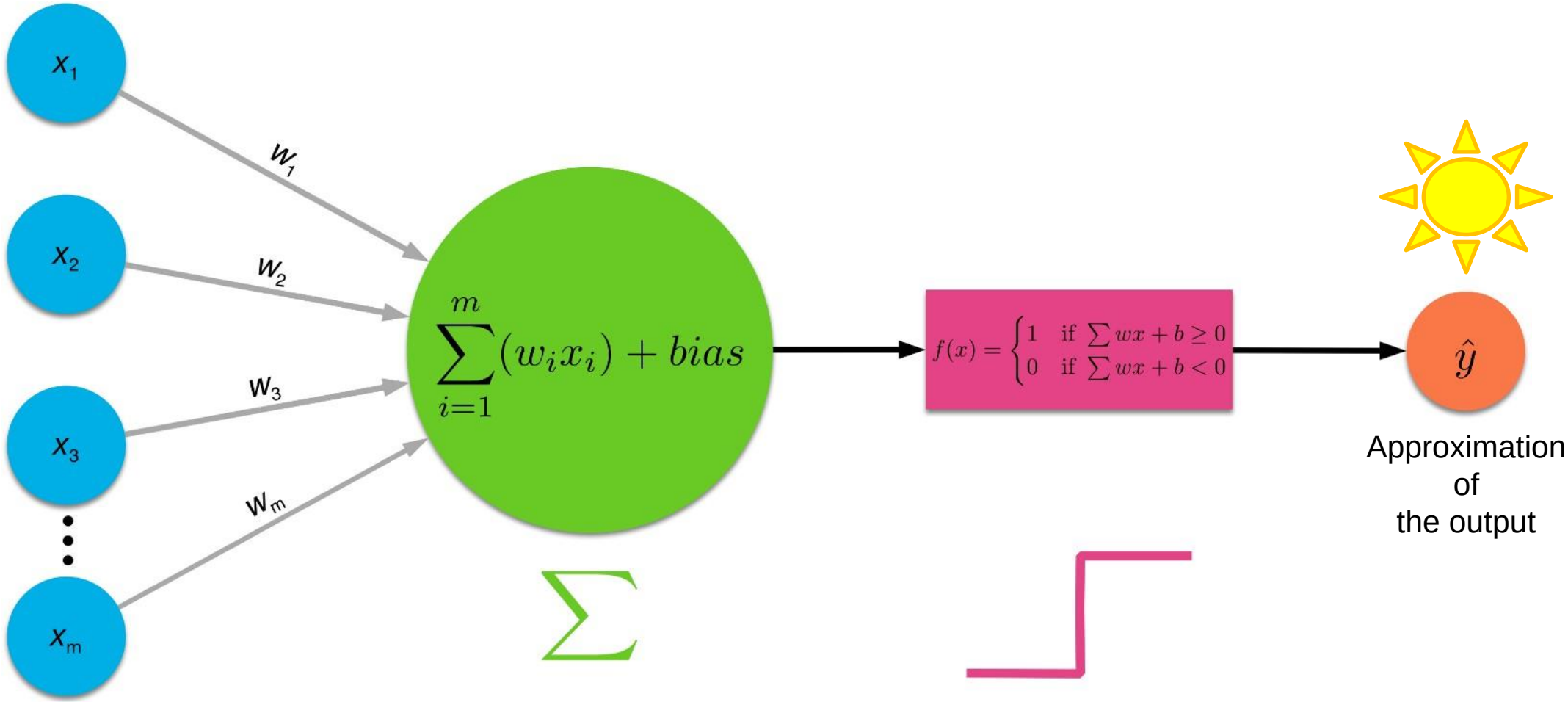
What are Neural Networks?

- Also referred to as **A**rtificial **N**eural **N**etworks (ANN)
- Inspired by biological neural system
- Biological neuron has three main components
  - **Dendrites** → "Inputs"
    - Take inputs from other neurons in terms of electrical pulses.
  - **Cell body** → "Processor"
    - Makes the inferences and decides the actions to take.
  - **Axon terminals** → "Output"
    - Send the outputs to other neurons in terms of electrical pulses.



Artificial Neuron

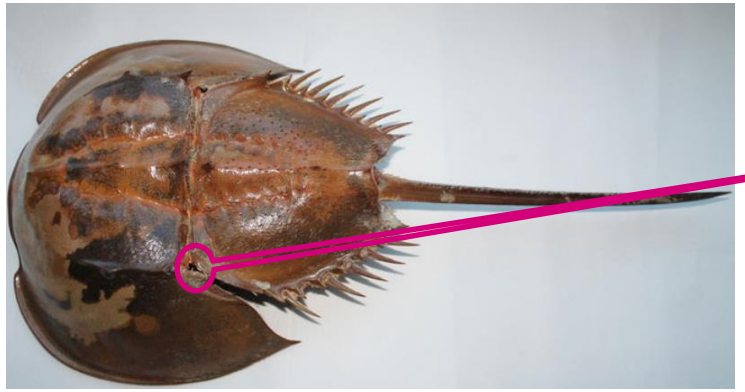
The heart of a neural network



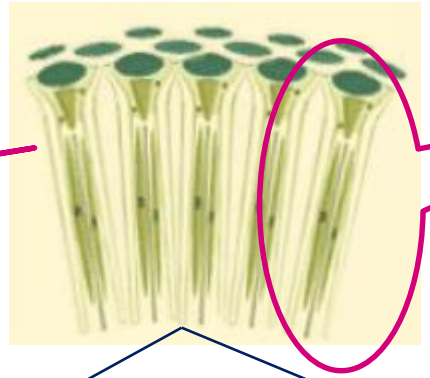
life.augmented

Limulus - real use case

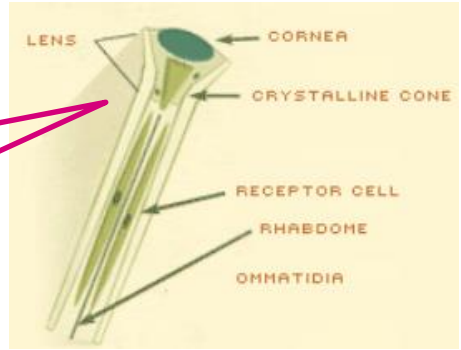
Limulus (Horseshoe) crab



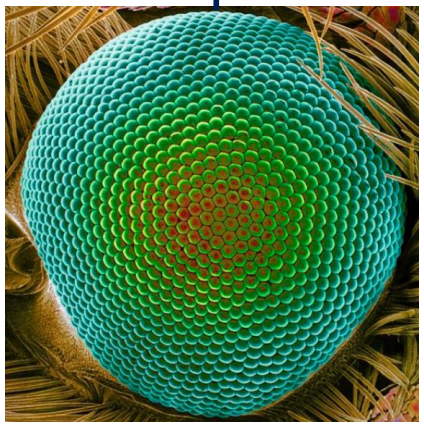
Compound eye



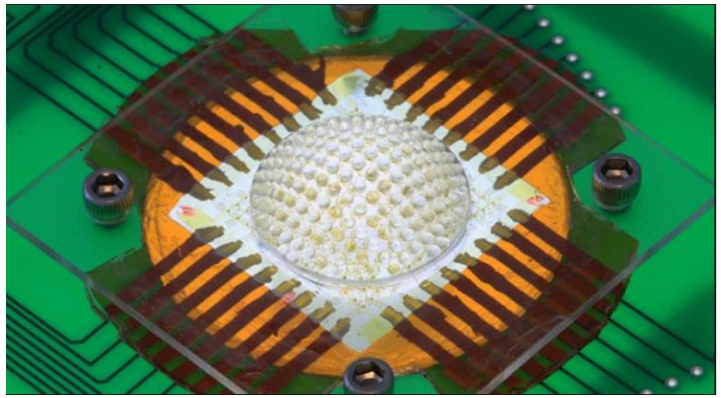
“Sensor” details



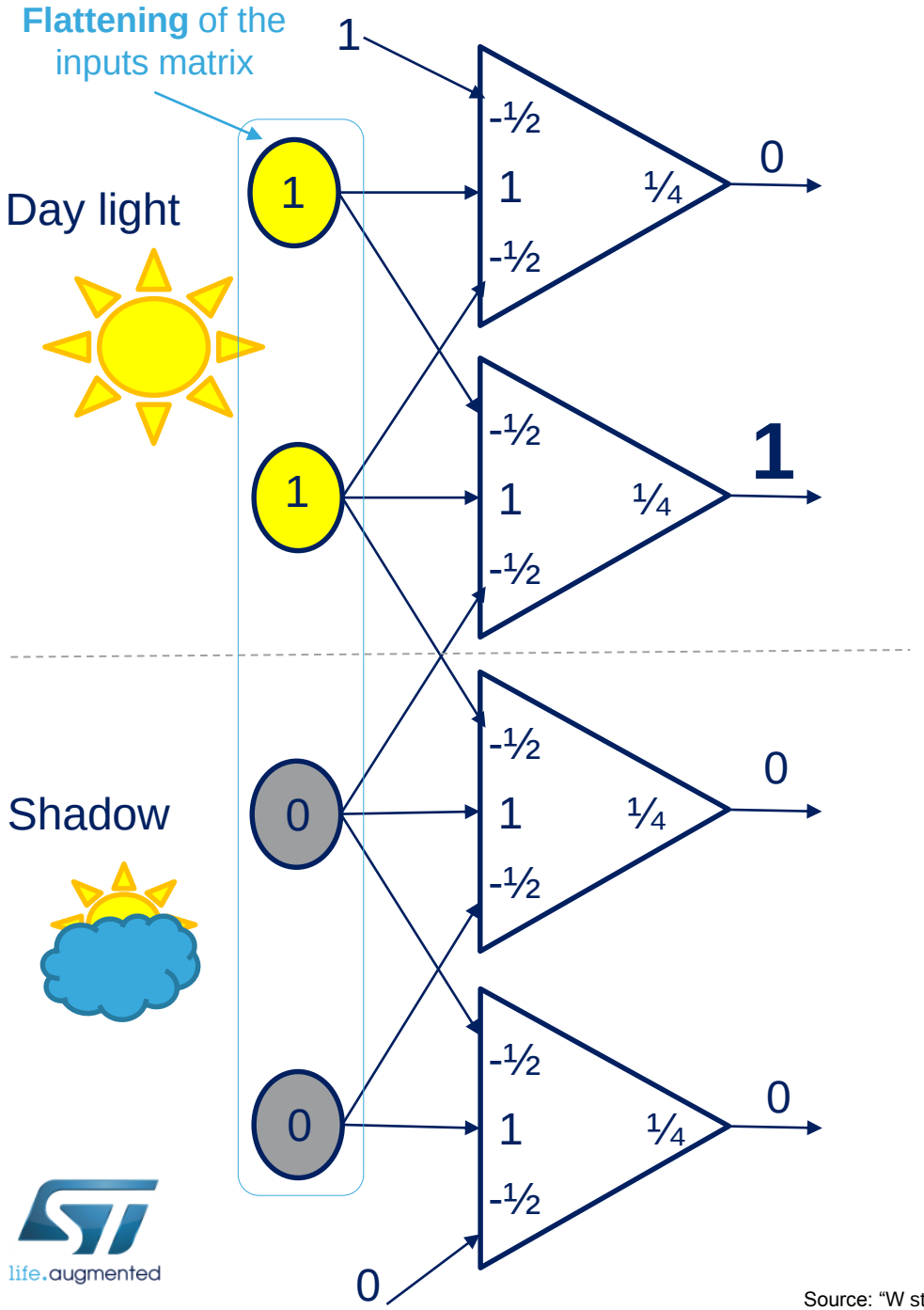
Insect compound eye



Artificial (IC) compound eye



Limulus – real use case



The structure of Limulus crab eyes neural network has been found as result of research.

Analyze Limulus crab eyes neural network behavior

- Crab can see nothing if is exposed to full day light !
- Crab can see border of light – shadow areas.
- Crab can easily detect moving objects.

Conclusion

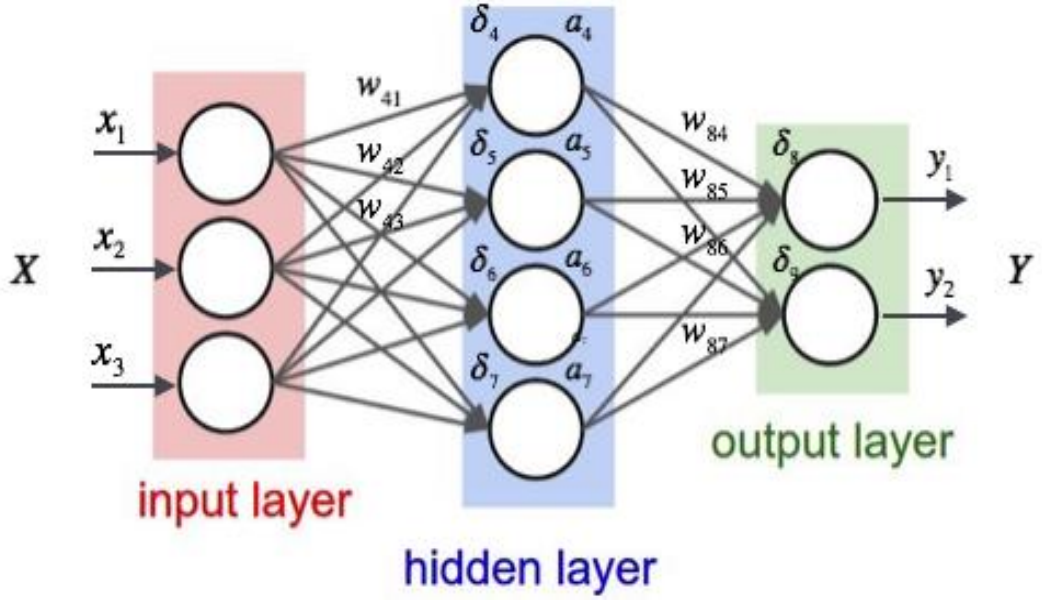
Biological neural network structure can be directly reused i.e. in image pre-processing software, the main advantage of implementation is significant reduction of data stream to process.

life.augmented

Source: “W strone usmiechnietych maszyn: spacer pograniczem biologii i techniki”, Ryszard Tadeusiewicz, Wydawinictwo ALFA, Warszawa 1989, ISBN 83-7001-231-0

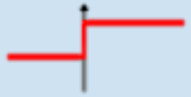
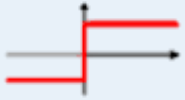
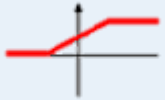
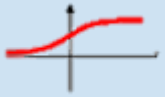
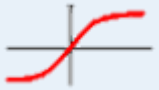
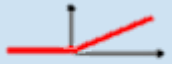
Layers of a Neural Network

- Neural network has three types of layers
- Input layer
  - Can be from other neurons or feature inputs
    - Age, height, weight, pixels in the images etc.
- Hidden layers (one or more)
  - Real power lies here
  - Adding more neurons to the network
- Output layer
  - Gives the output we want to predict
    - Probability of rain
    - Object class
    - Disease is fatal or not…



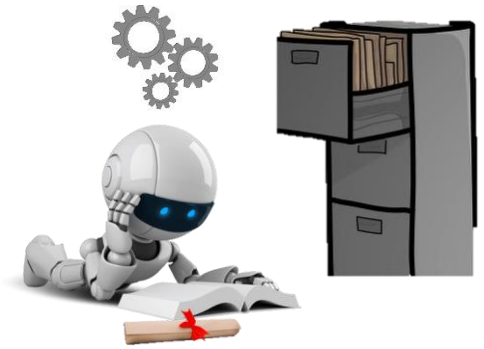
$$z = \sum_{i=1}^{m} w_i x_i + bias$$

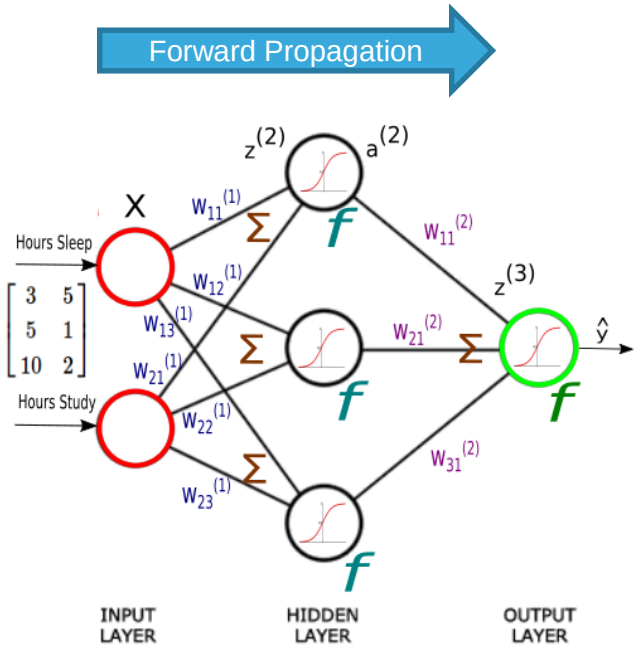
Activation Functions

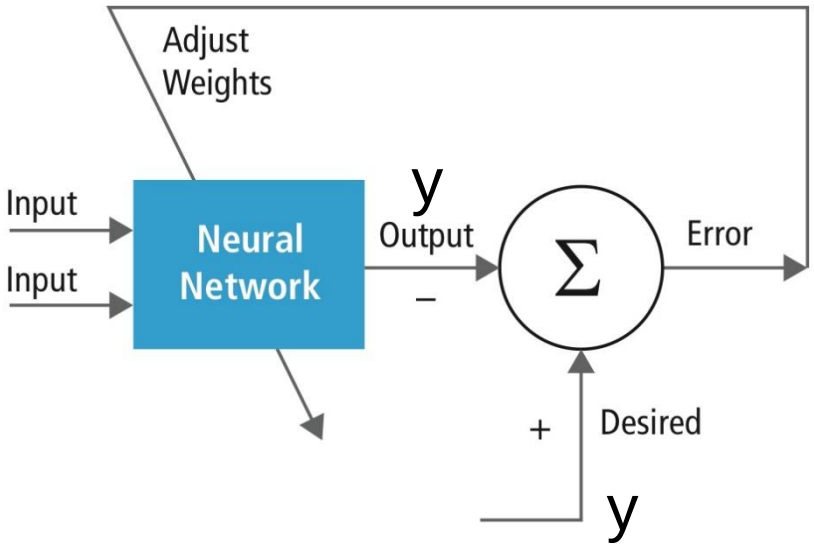
Name	Plot	Function	Examples
Unit Step		$\emptyset(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant
Sign (Signum)		$\emptyset(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant
Linear		$\emptyset(z) = z$	Adaline, linear regression
Piece-wise linear		$\emptyset(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z0 < \frac{1}{2}, \\ 1, & z > 0, \end{cases}$	Support vector machine
Logistic (sigmoid)		$\emptyset(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi layer-neural networks
Hyperbolic Tangent		$\emptyset(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi layer neural networks
Rectified Linear Unit (ReLU)		$\emptyset(z) = \begin{cases} z, & z > 0 \\ 0, & z \leq 0 \end{cases}$	Regression, approximation, multi layer neural network

Training neural networks

- In supervised learning we have a relatively large dataset.
- Feed all the samples as inputs to get an output. Called forward propagation.
- At start the weights can be randomized or predefined depending on the applications scenario.
- The result $\hat{y}$ is compared with actual output y.
- The task of course is to make the output value $\hat{y}$ to be as close to y as possible reducing the error.

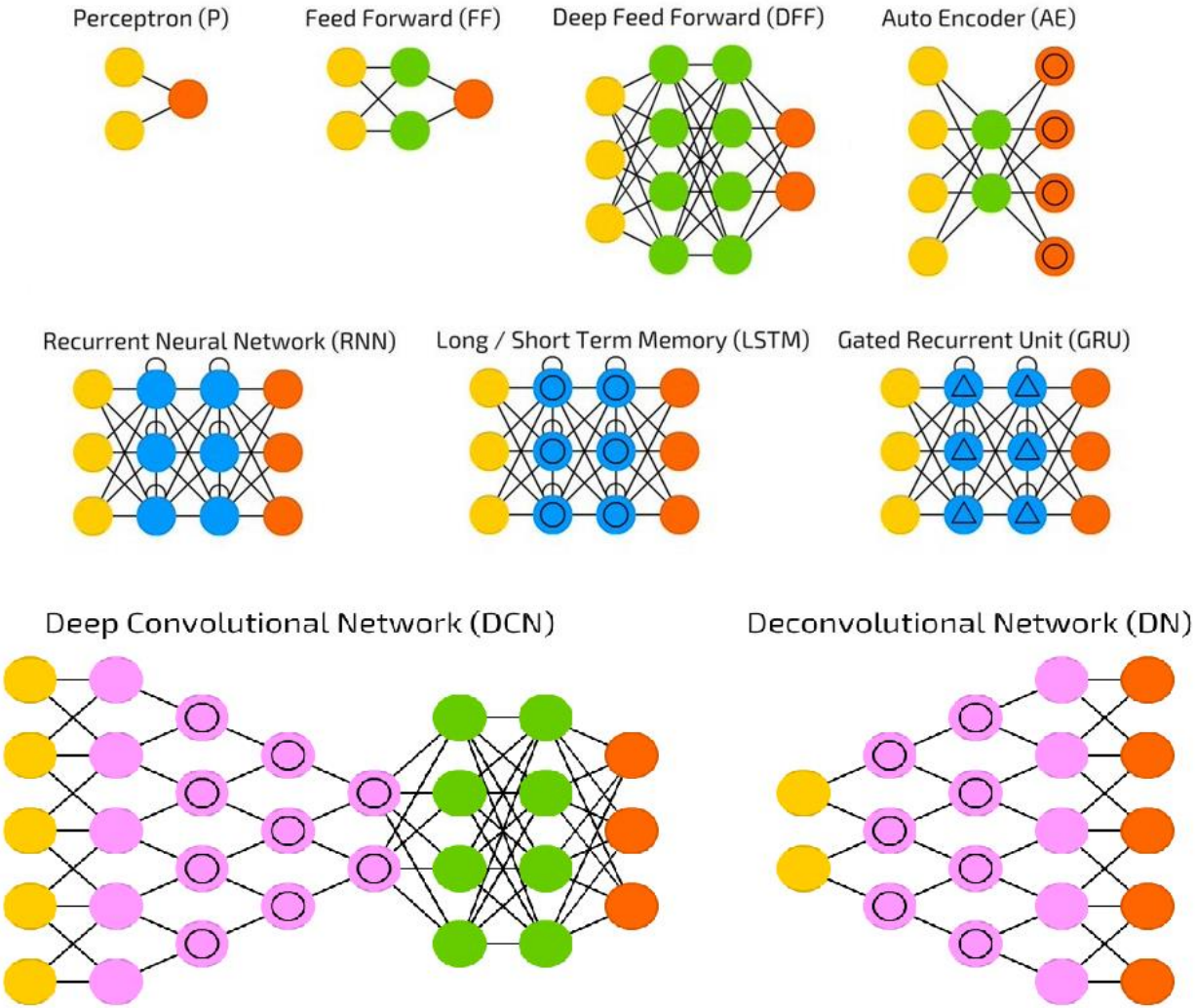






life.augmented

Neural Networks Types



"Out-of-the-box" experience

How to **program** and **run precompiled binaries** on the **STM32L4 Discovery kit IoT node**

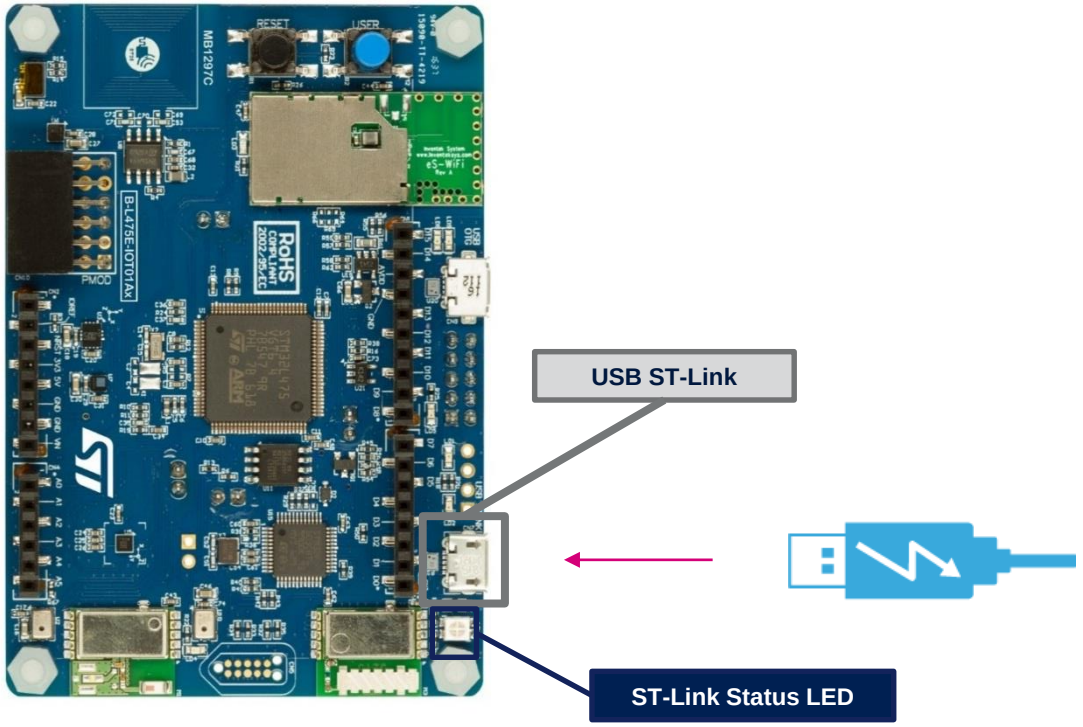
Key learning

- To be familiar with FP-AI-SENSING1 Audio Scene Classification ANN application functionality

Lab 1: *"Out-of-the box"* experience

Step 1 - Connect the board

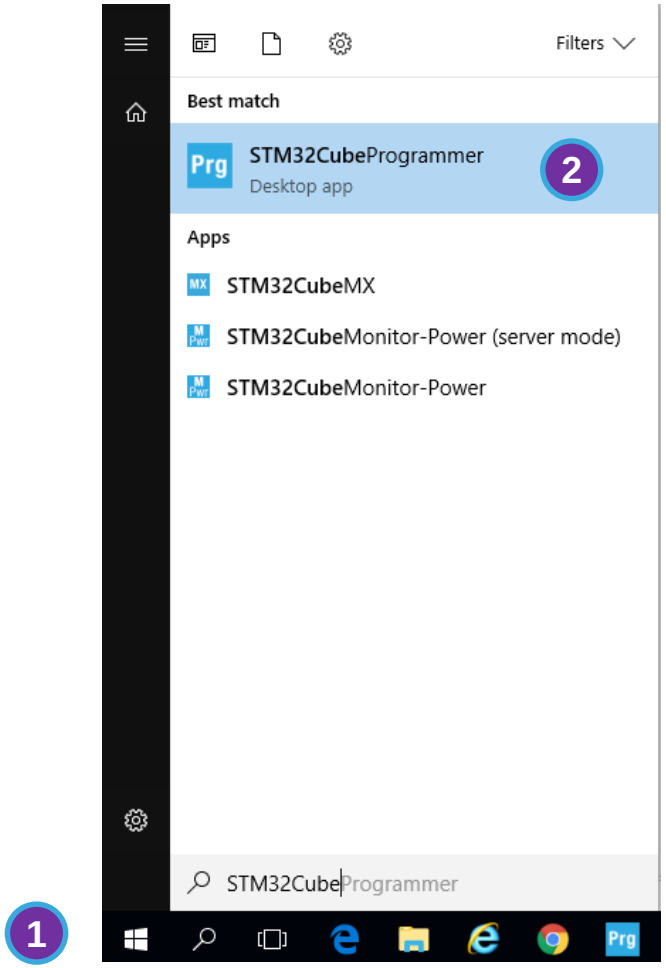
- Connect your PC to the USB ST-Link.
- The board will be powered through the ST-Link connection.
- The ST-Link Status LED will be steady when ST-Link is recognized.



Lab 1: *"Out-of-the box"* experience

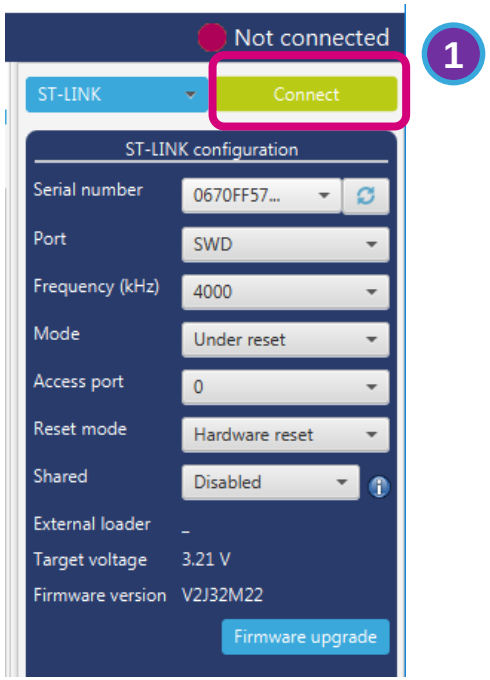
Step 2 - Program the board (1/3)

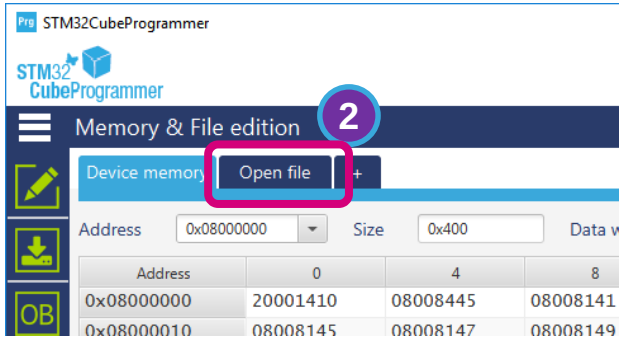
- Launch STM32CubeProgrammer



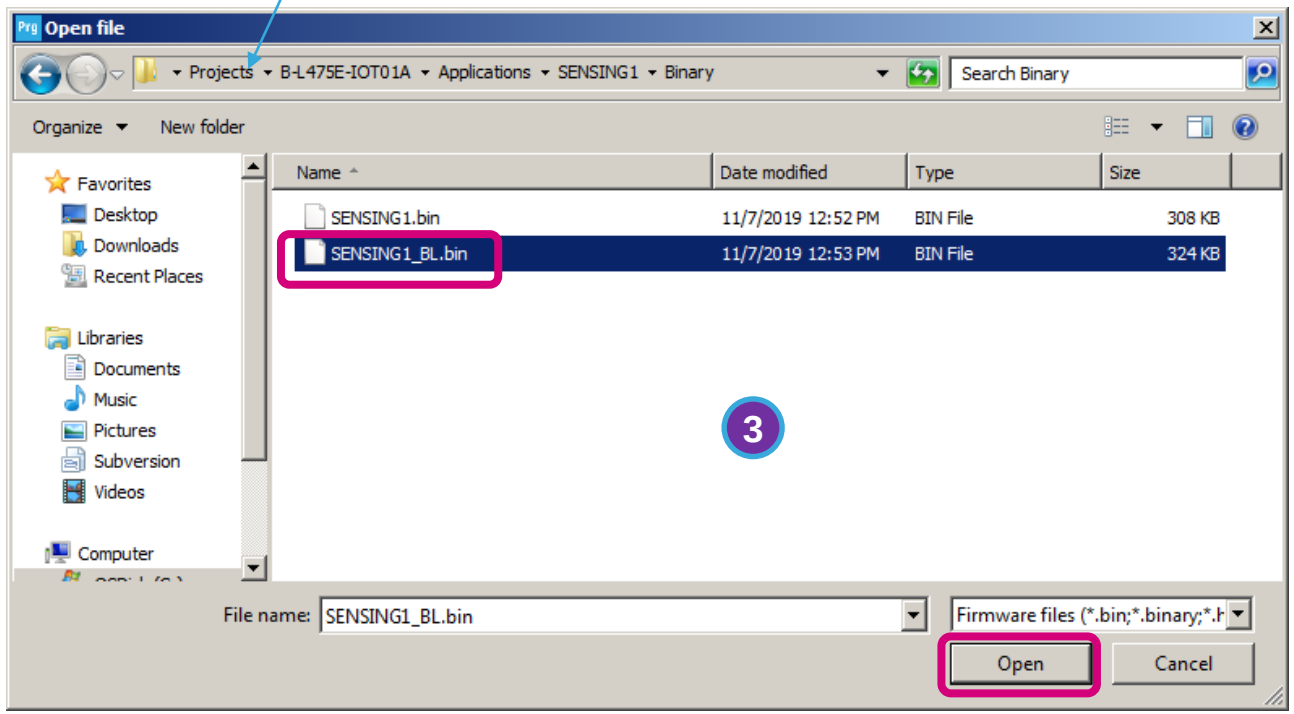
Lab 1: *"Out-of-the box"* experience

Step 2 - Program the board (2/3)





C:\AI\FP-AI-SENSING1\
Projects\
B-L475E-IOT01A\
Applications\
SENSING1\
Binary\

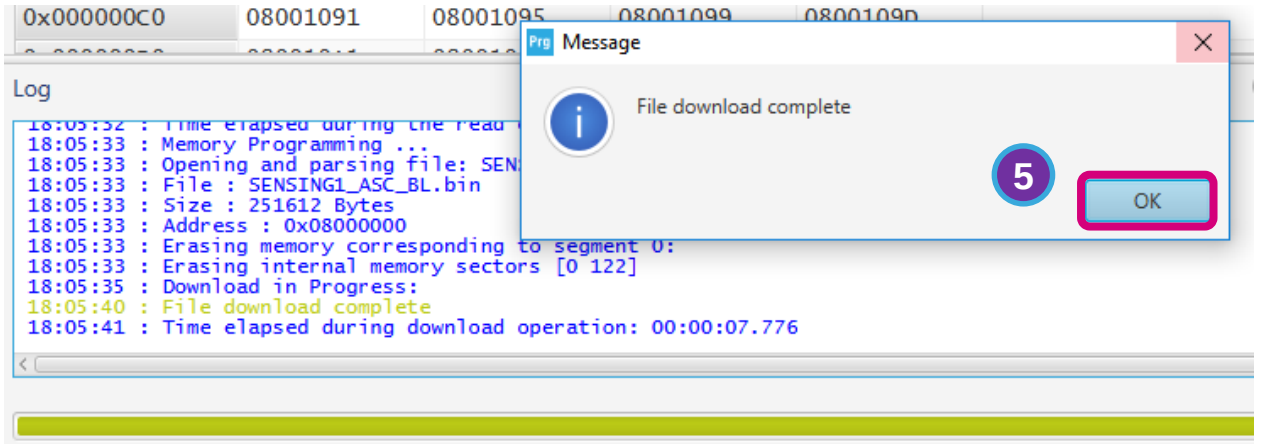


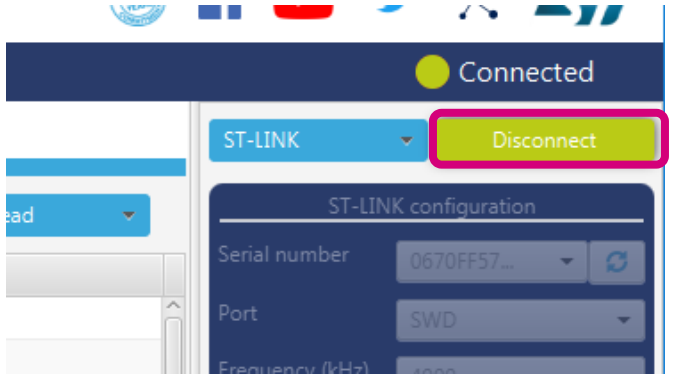
If boards fails to connect, check target has latest ST-Link Firmware

life.augmented

Lab 1: *"Out-of-the box"* experience

Step 2 - Program the board (3/3)

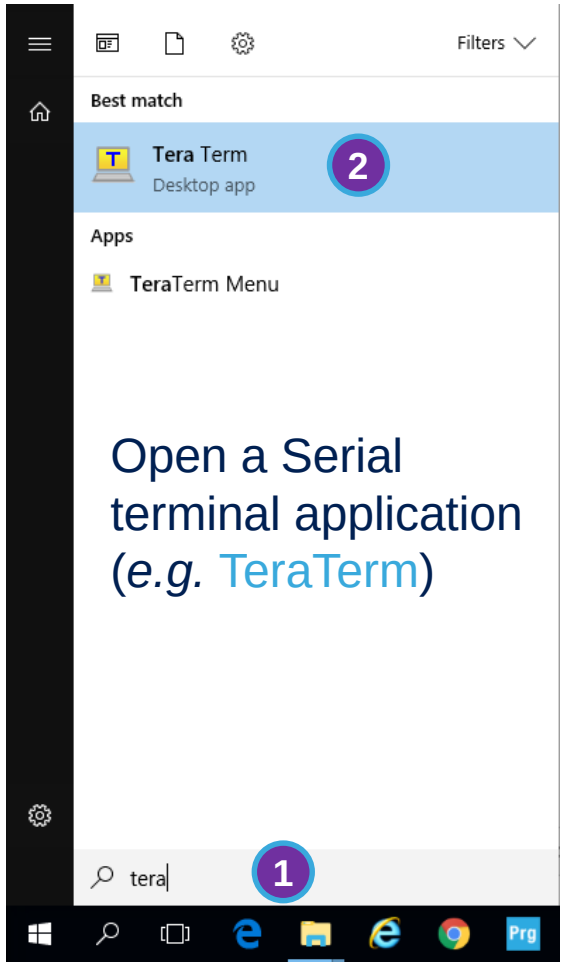




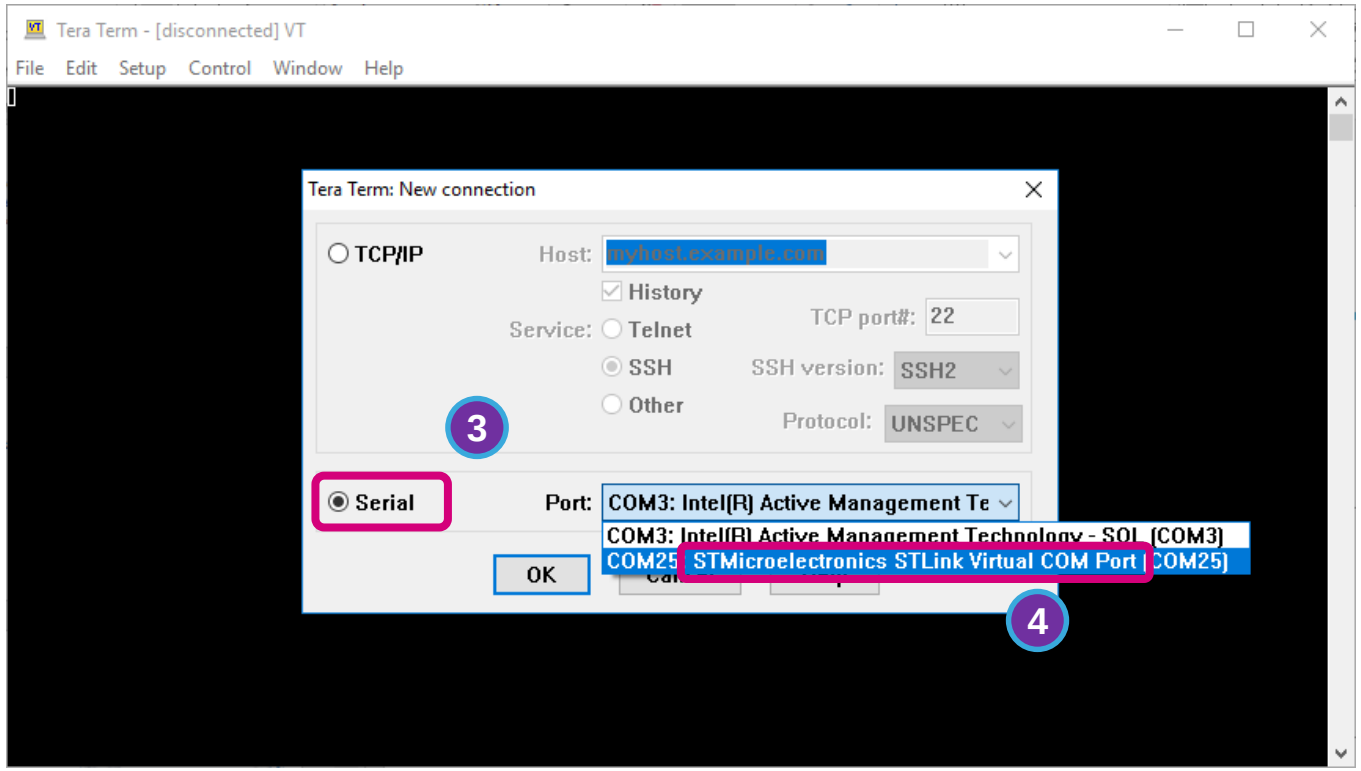
life.augmented

Lab 1: *"Out-of-the box"* experience

Step 3 – UART Terminal (1/3)



Select Serial Port: COMXX: STMicroelectronics STLink Virtual COM Port

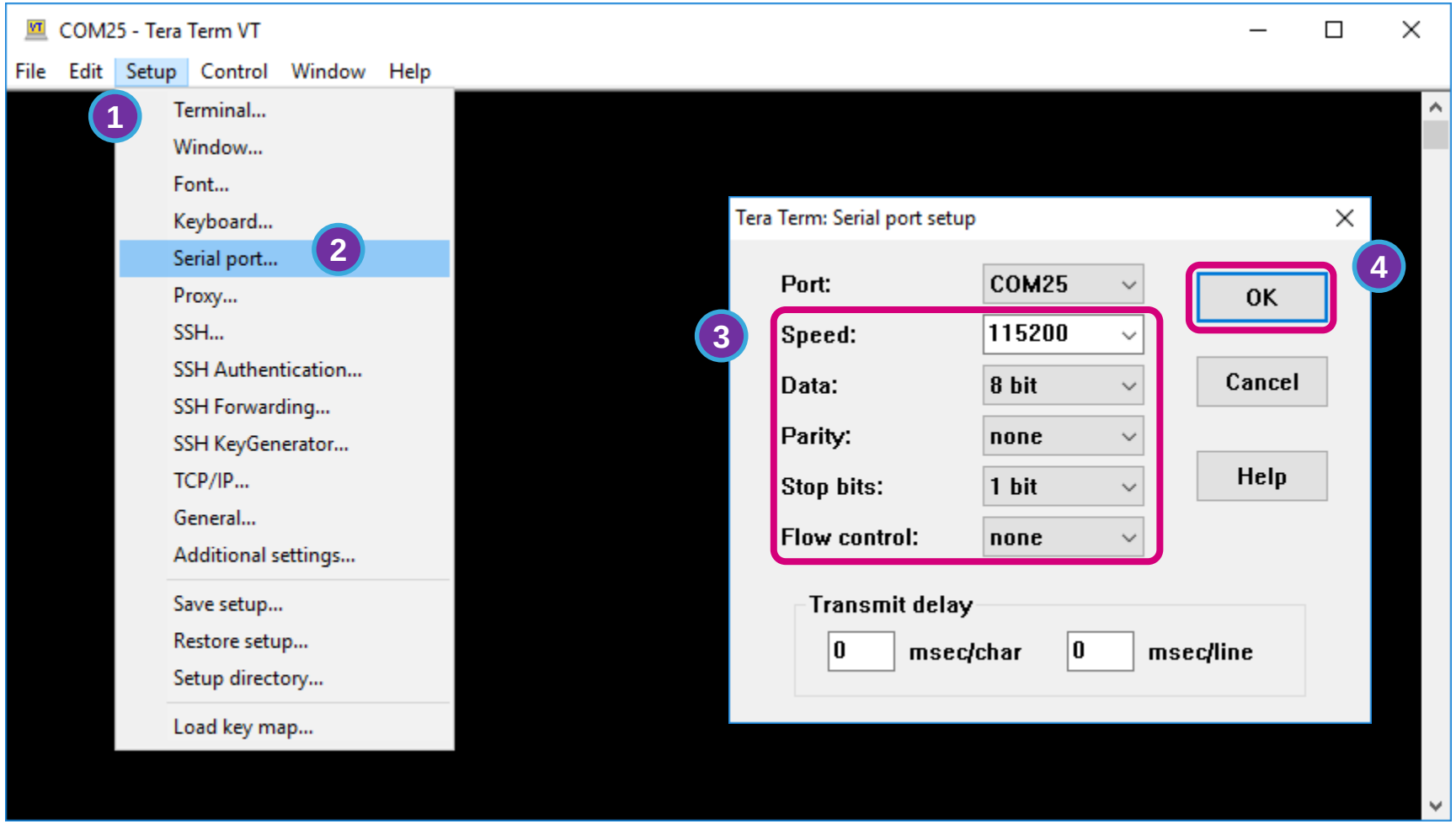


life.augmented

Lab 1: *"Out-of-the box"* experience

Step 3 – UART Terminal (2/3)

Set serial port parameters to 115200 baud, 8-bit, no parity, 1 stop bit

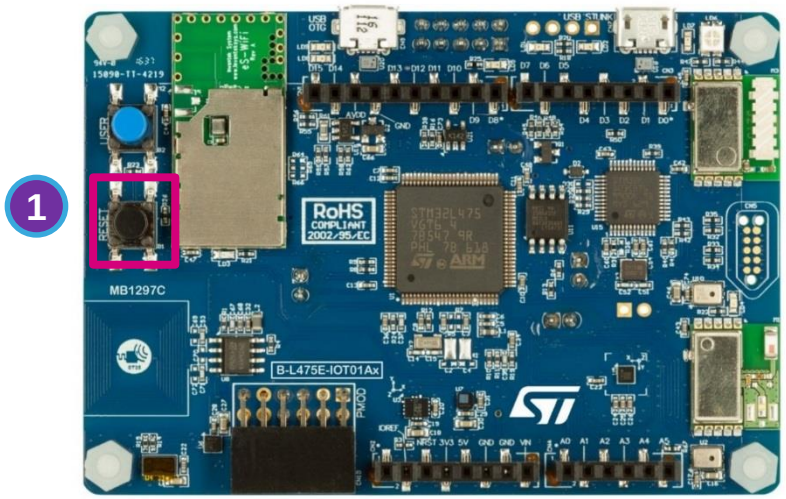


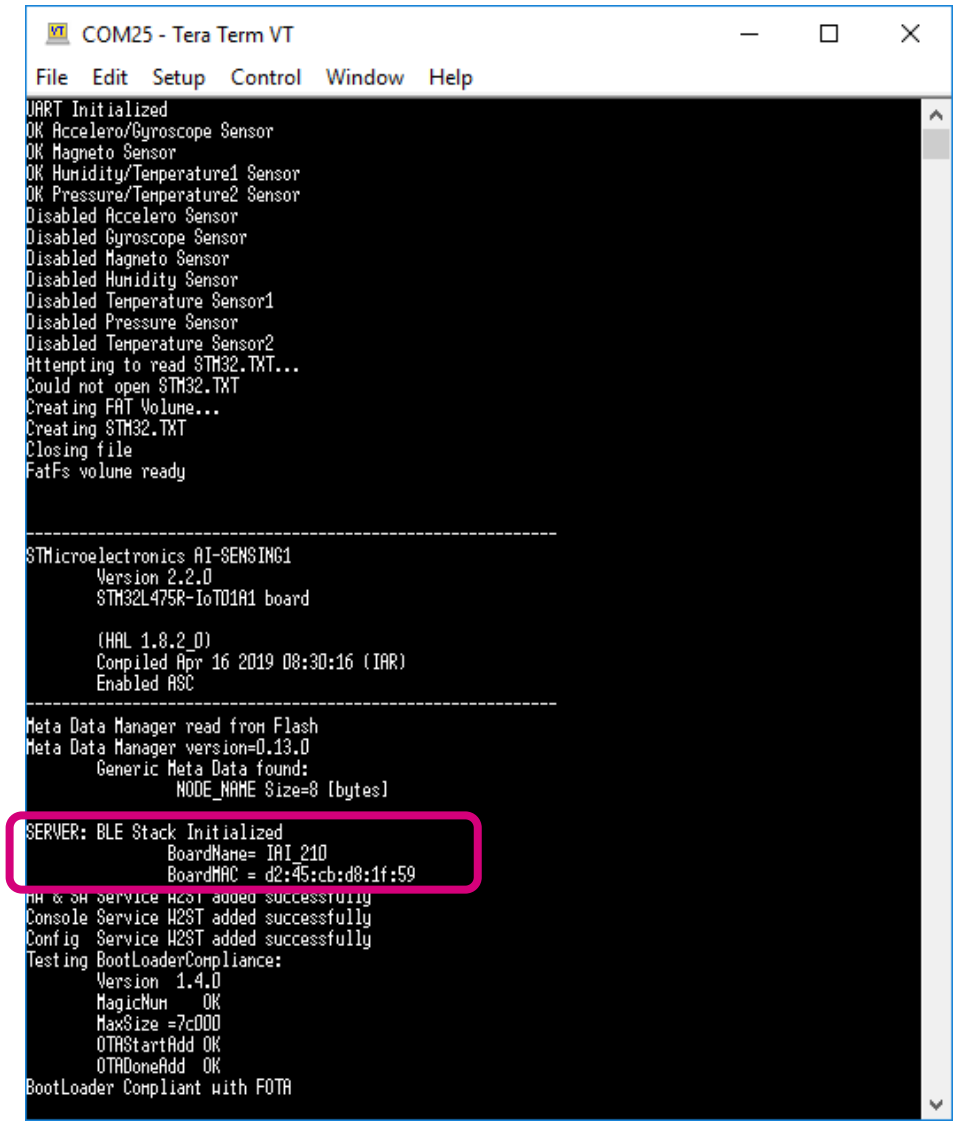
Lab 1: *"Out-of-the box"* experience

Step 3 – UART Terminal (3/3)

- Reset the board (press the **black** RESET button)
- Identify your board BLE MAC address in the log. *E.g.*:

```
SERVER: BLE Stack Initialized
                BoardName= IAI_300
                BoardMAC = d2:45:cb:d8:1f:59
```



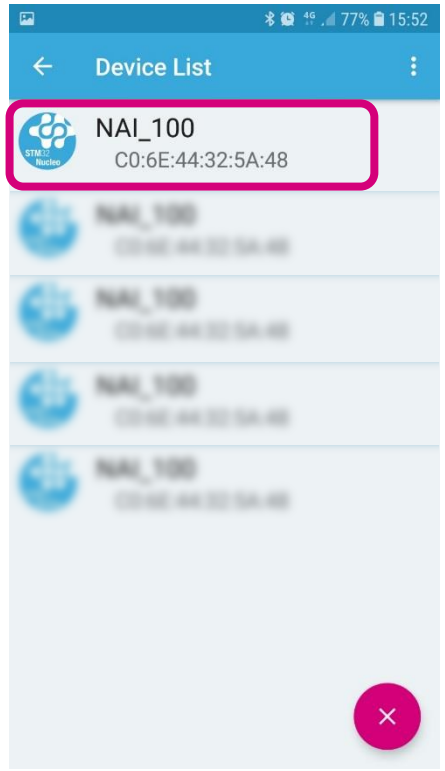


life.augmented

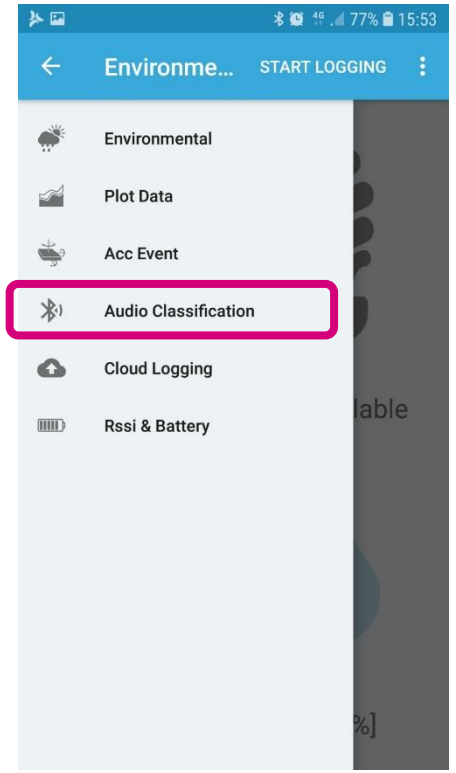
Lab 1: *"Out-of-the box"* experience

Step 3 – Connect to Smartphone

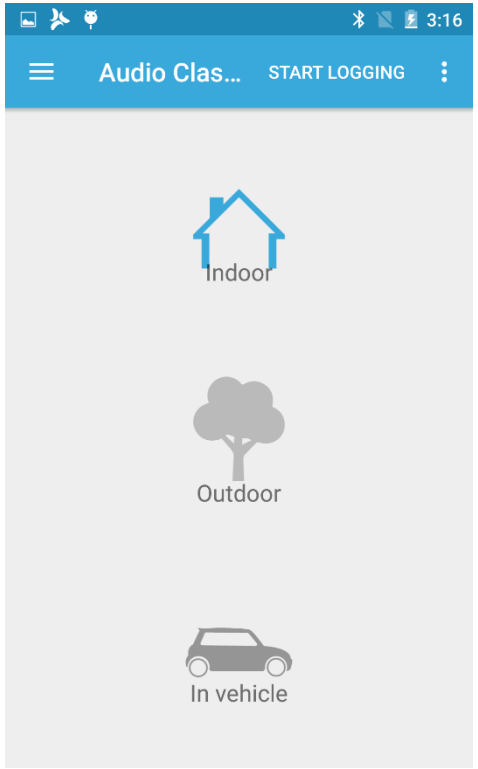
Starting Page



Discovery Page



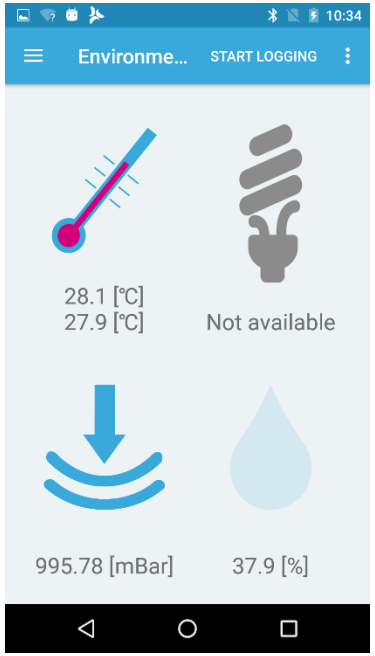
Device Connected



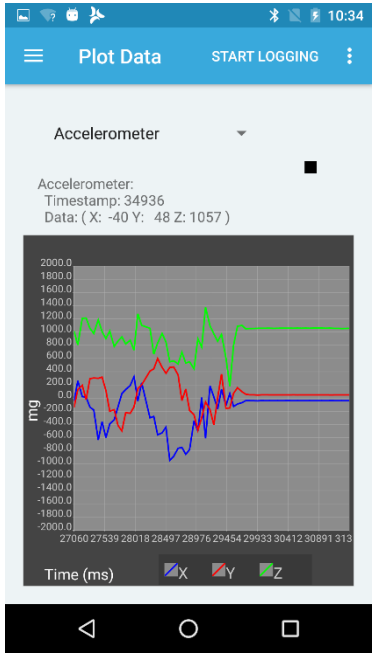
Audio Scene Classification

life.augmented

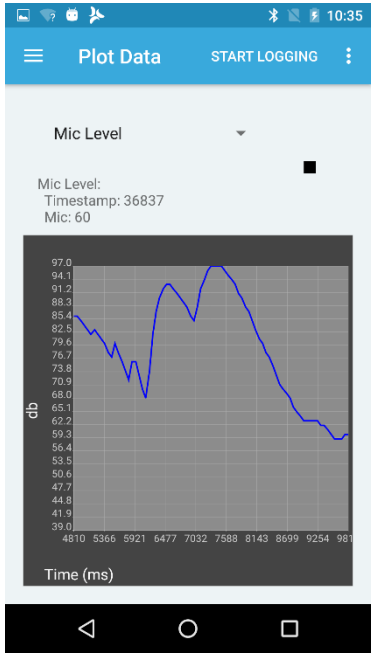
ST BLE Sensor Application for **Android/iOS** Overview



Environmental

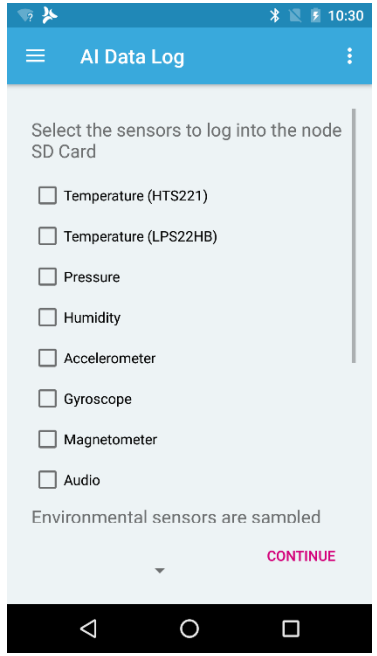


Accelerometer plot

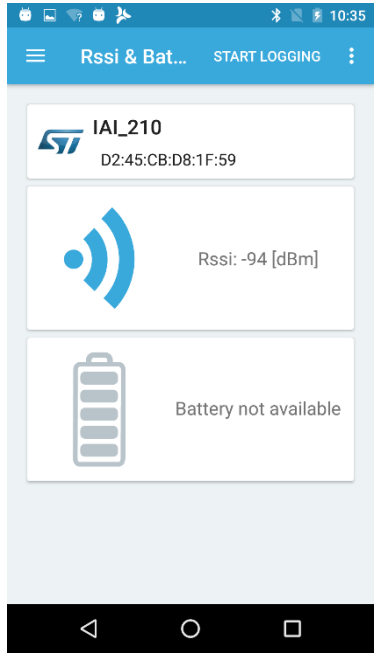


Microphones level plot

Audio Classification



AI Data log



RSS & Battery

life.augmented

Neural Networks on STM32

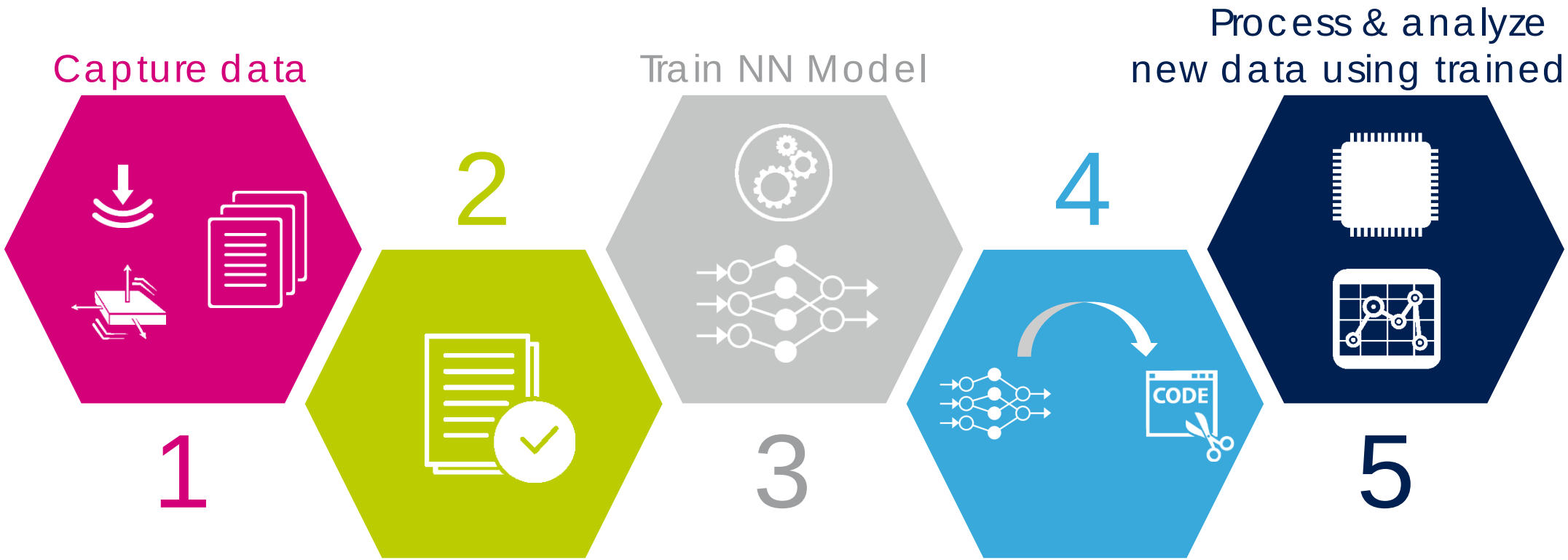
Simple, fast, optimized

Five steps behind ANN

STM32 Cube.AI

The Key Steps Behind Neural Networks

Neural Network (NN) Model Creation | Operating Mode



life.augmented

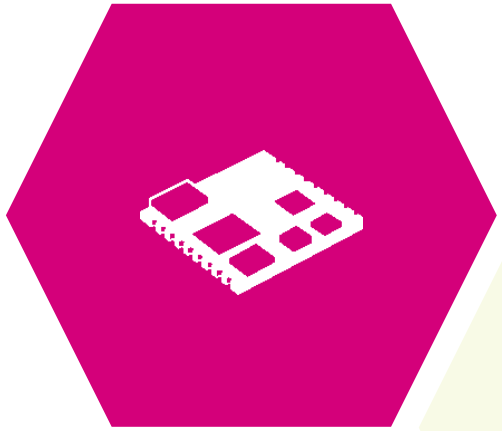
ST Toolbox for Neural Networks

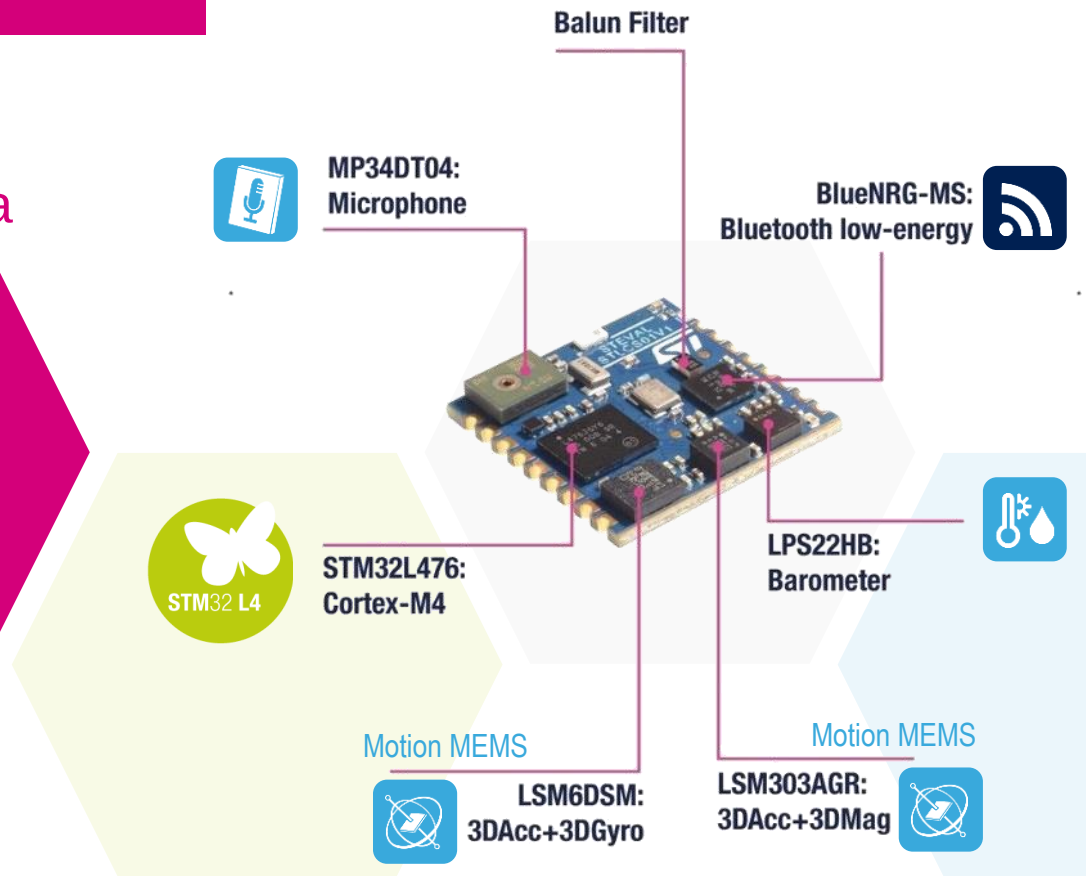
Form Factor Hardware
to Capture and Process Data

Capture Data | Label Data | Train NN | STM32 Cube.AI | Run on STM32

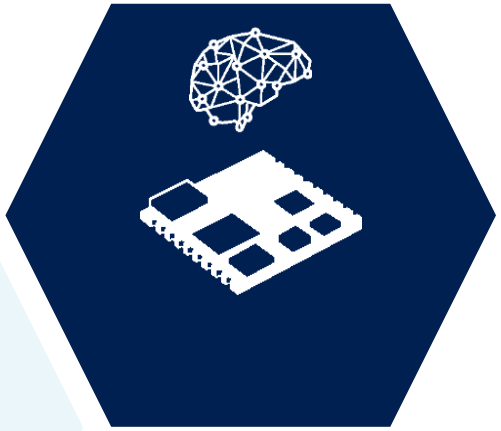
SensorTile

Capture data





Process & analyze
new data using trained NN



www.st.com/SensorTile
www.st.com/SensorTile-edu

life.augmented

Fast Go to Market Module

to Capture Data with More Accuracy

Capture Data | Label Data | Train NN | STM32 Cube.AI | Run on STM32

SensorTile.Box

Capture data

Process & analyze new data using trained NN

Microsoft IoT Services ready

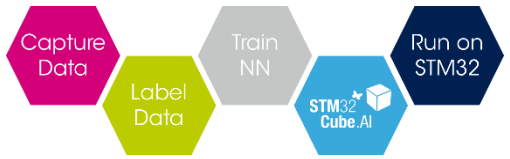
Microsoft Azure

More advanced, high accuracy and low power sensors

- First Inertial module with Machine Learning capabilities.
- Motion (accelerometer and gyroscope, magnetometer) and slow motion (inclinometer)
- Altitude (pressure), environment (pressure, temperature, humidity, compass) and sound (sound and ultrasound analog microphone)
- Microsoft IoT services ready to make available on a web dashboard the result of the embedded processing

www.st.com/SensorTileBox

life.augmented



Distributed AI: Sensor + STM32

Optimize Performance and Power Consumption

Smart Sensor
with Machine Learning Core

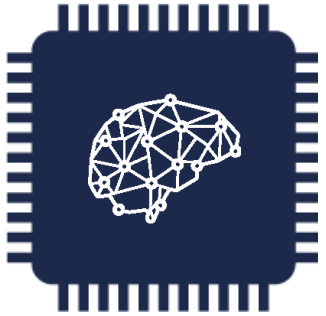
Smart STM32
second level of AI processing

Inertial Sensor
New LSM6DSOX

Raw Data

Event Decision

FSM and MLC
Re-configuration



Deep Learning

Neural Networks

Machine Learning

- Best ultra-low-power sensing at high performance:
  - 550µA (gyroscope and accelerometer)
    ➔ 200µA less than closest competitor
  - 20~40µA (Accelerometer only for HAR)
- Efficient Finite State Machines: 2µA
- Configurable Machine Learning Core: 4~8µA

- More advanced and complex NNs
- Decisions on multiple sensors
- NN input can be sensor data and/or sensor Machine Learning decisions
- Multiple Neural Networks support
- Actuation & communication

life.augmented

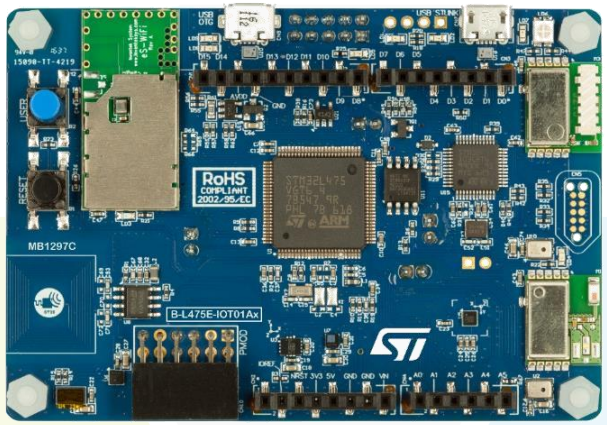
Form Factor Hardware

AI IoT Node for More Connectivity

IoTNode

\+ Sub-1GHz Dynamic NFC Tag Wi-Fi

Capture data

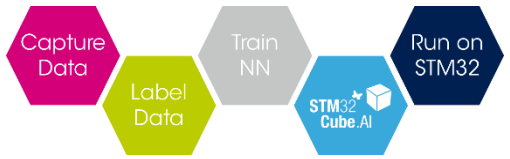


Process & analyze new data using trained NN

More debug capabilities

- Integrated ST-Link/V2.1
- PMOD extension connector
- Arduino Uno extension connectors

https://www.st.com/IoTnode



Collecting Data & Architecting a NN Topology

Services provided by Partners

Capture data

Clean, label Data
Build NN topology

ST tools to support

ST BLE Sensor

ST BLE Sensor mobile phone application

Collect and label data from the SensorTile.

life.augmented

Partner Program

Selected partners

Neural Networks engineering services support.

Data scientists and Neural network architects.

life.augmented

STM32CubeMX Extension

AI Conversion Tool

Capture Data · Label Data · Train NN · STM32 Cube.AI · Run on STM32

Input your framework-dependent, pre-trained Neural Network into the **STM32Cube**.AI conversion tool

Automatic and fast generation of an STM32-optimized library

STM32Cube.AI offers interoperability with state-of-the-art Deep Learning design frameworks

Train NN Model

Keras · TensorFlow* · ConvNetJS Deep Learning in your browser · Lasagne · Caffe

* TensorFlow used as a Keras backend. Not all operators accessible to MCUs

STM32 Cube.AI

Convert NN into optimized code for MCU

Process & analyze new data using trained NN

STM32

life.augmented

STM32Cube.AI Roadmap

2019

- Floating Point Support

Keras
ConvNetJS
Deep Learning in your browser
Lasagne
Caffe

Apr

IoTNode

SensorTile.Box

Jun

- Quantization
- TensorFlow for MCU
- Command line interface
- UI Improvements
- Additional layers

\+ TensorFlow Lite microcontrollers

2020

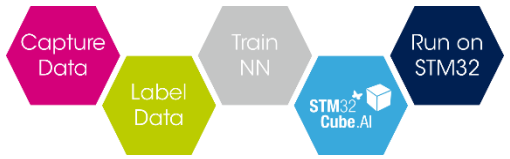
- ONNX introduction
- Additional layers
- Debug improvements

Keras + ONNX mxnet Microsoft CNTK Chainer PyTorch







ST Toolbox for Neural Networks

More Than Just a Conversion Tool

Resources

- Function packs for **quick prototyping**
- **Audio** and **Motion** examples

- STM32 **Community** with **dedicated** Neural Networks **topic**
- For **support** and **ideas** exchange

Process & analyze
new data using trained NN

Convert NN into
optimized code for MCU

life.augmented

STM32 Function Packs

STM32 Function Packs are a combination of low-level drivers, middleware libraries and sample applications assembled into a single software package.

Function Packs **help jump-start** the implementation and the development of a number of “functions” - for example a gateway or a node in a wireless sensor network – in different domains.

☐	FP-IND-PREDMNT1 ACTIVE	STM32Cube function pack for multi sensors node with signal processing to enable predictive maintenance	ST
☐	FP-AI-SENSING1 ACTIVE	STM32Cube function pack for ultra-low power IoT node with artificial intelligence (AI) application based on audio and motion sensing	ST
☐	FP-ATR-LORA1 ACTIVE	STM32Cube function pack for IoT tracker node with LoRa connectivity, GNSS and sensors	ST
☐	FP-ATR-SIGFOX1 ACTIVE	STM32Cube function pack for IoT tracker node with Sigfox connectivity and sensors	ST
☐	FP-ATR-TOMTOM1 ACTIVE	STM32Cube function pack for IoT node with GNSS and cellular connectivity for Asset Tracking applications based on TomTom online services	ST
☐	FP-AUD-BVLINK1 ACTIVE	STM32 ODE function pack for half-duplex voice streaming over Bluetooth low energy	ST
☐	FP-AUD-BVLINK2 ACTIVE	STM32Cube function pack for full-duplex voice streaming over Bluetooth low energy using Opus compression	ST

life.augmented

https://www.st.com/en/embedded-software/stm32-ode-function-pack-sw.html#products

FP-AI-SENSING1 Function Pack Scope

- FP-AI-SENSING1 is a STM32Cube function pack which contains drivers, middleware and sample application to provide use cases based on Artificial Neural Network libraries generated by X-CUBE-AI
- Educational Artificial NN trained on a limited dataset.
  Goal is that our customers integrate easily their own NN.
- Simple setup based on STM32 Nucleo or SensorTile development boards
- Includes power optimizations
- Companion “ST BLE Sensor” Android/iOS app allowing users to evaluate Neural Networks outputs, applied to Human Activity Recognition (HAR) and Acoustic Scene Classification (ASC)

FP-AI-SENSING1

Application — FP-AI-SENSING1

Middleware
- USB Device
- FatFS
- BLE
- FOTA
- Power Controller
- FreeRTOS
- Sensing pre-proc:gravity, Rodrigues
- Audio pre-proc: log Mel
- AI NN Library

Hardware Abstraction — STM32Cube Hardware Abstraction Layer (HAL)

Hardware
- STM32 Nucleo expansion boards
  - X-NUCLEO-IDB05A1 (Connect)
  - X-NUCLEO-IKS01A2 (Sense)
  - X-NUCLEO-CCA02M1 (Sense)
- STM32 Nucleo L4/F4 development board
- STEVAL-SLTKT01V1 evaluation board

Motion sensors

Environmental sensors

Audio Microphones

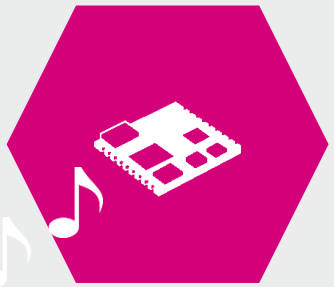
STM32 Cube.AI

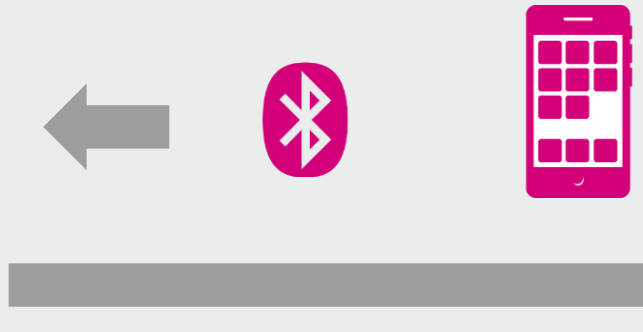
- **Human Activity Recognition**
- **Acoustic Scene Classification**

life.augmented

Audio Scene Classification (ASC)

Audio Example in FP-AI-SENSING1 Package





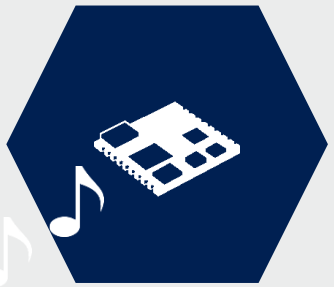
3 classes

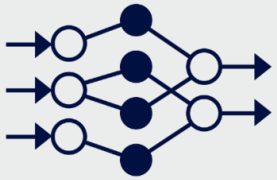
Indoor, Outdoor, In vehicle

Embedded **audio**

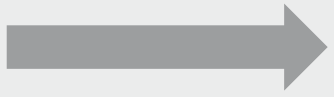
Labelling controlled
by smartphone application

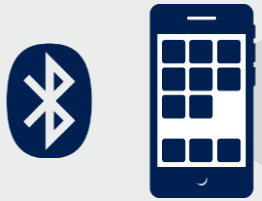
Data stored on the device
SD card for future **learning**





NN & example
dataset provided





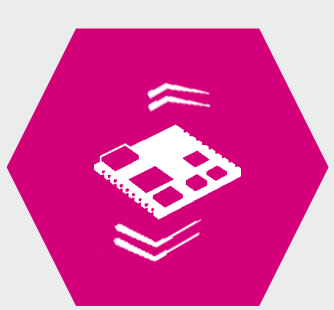
Embedded **audio**
pre-processing

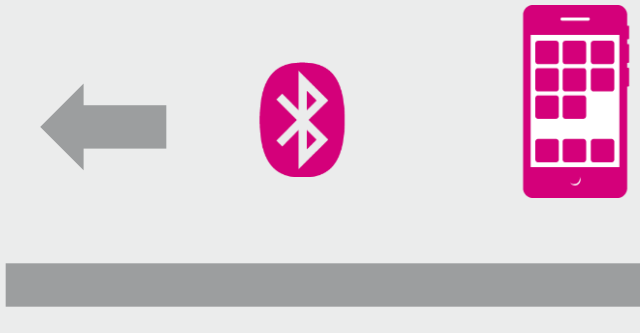
Inferences running
on the microcontroller

Inference result
displayed on mobile app

Human Activity Recognition (HAR)

Motion Example in FP-AI-SENSING1 Package





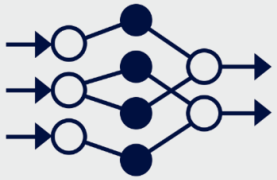
5 classes

Embedded **motion**

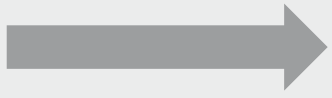
Labelling controlled
by smartphone application

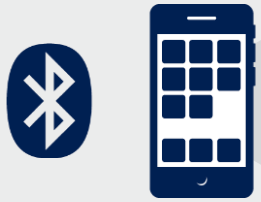
Data stored on the device
SD card for future **learning**

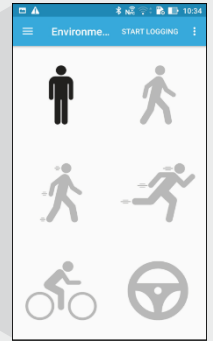
Stationary, walking, running,
biking, driving



NN & example
dataset provided





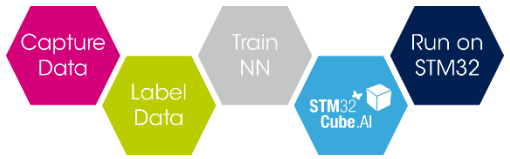


Embedded **motion**
pre-processing

Inferences running
on the microcontroller

Inference result
displayed on mobile app

life.augmented



STM32 Solutions for AI

More Than Just the STM32Cube.AI

An extensive toolbox to support easy creation of your AI application

AI extension for STM32CubeMX

To map pre-trained Neural Networks onto the STM32

STM32 Cube.AI

Function packs for Quick prototyping

Audio and motion examples

Reference hardware

To run inferences or data collection

… And more coming!

STM32 Community with dedicated Neural Networks topic

Mobile phone application

To collect and label data

To display the result of inference processing on the STM32

ST Partner Program with a dedicated group of Partners providing Neural Networks engineering services

Data scientists and Neural network architects

life.augmented

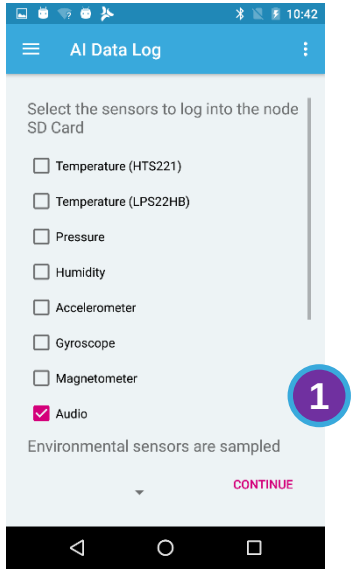
How to **collect** and **label** training dataset **?**

Key learning

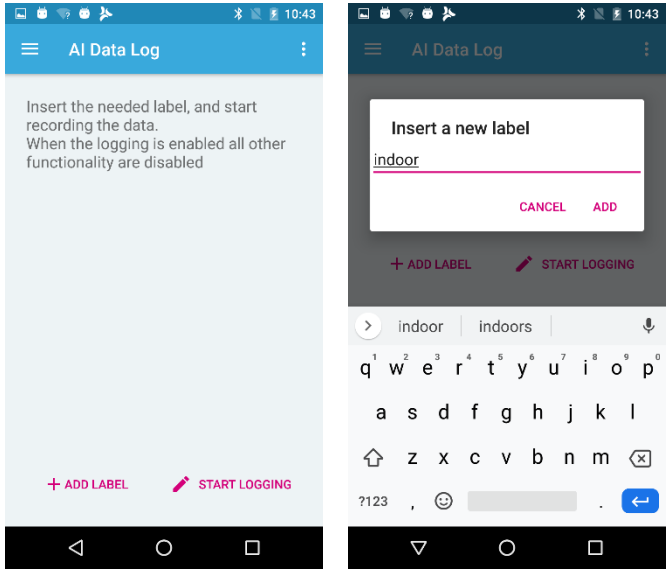
- Collection and labeling of training (input / expected output) data set of ANN

Lab 2: Data Collection

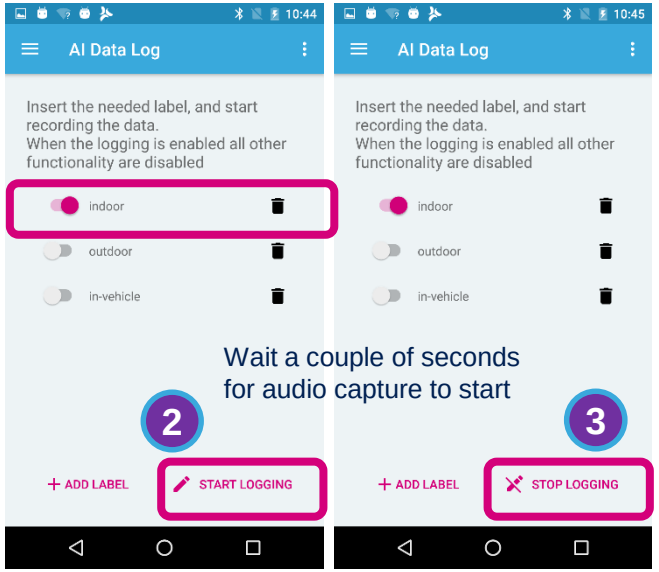
Step 1 – Audio Data Logging



Select Sensor(s)



Add labels for annotation (optional)



Log & annotate sensor data

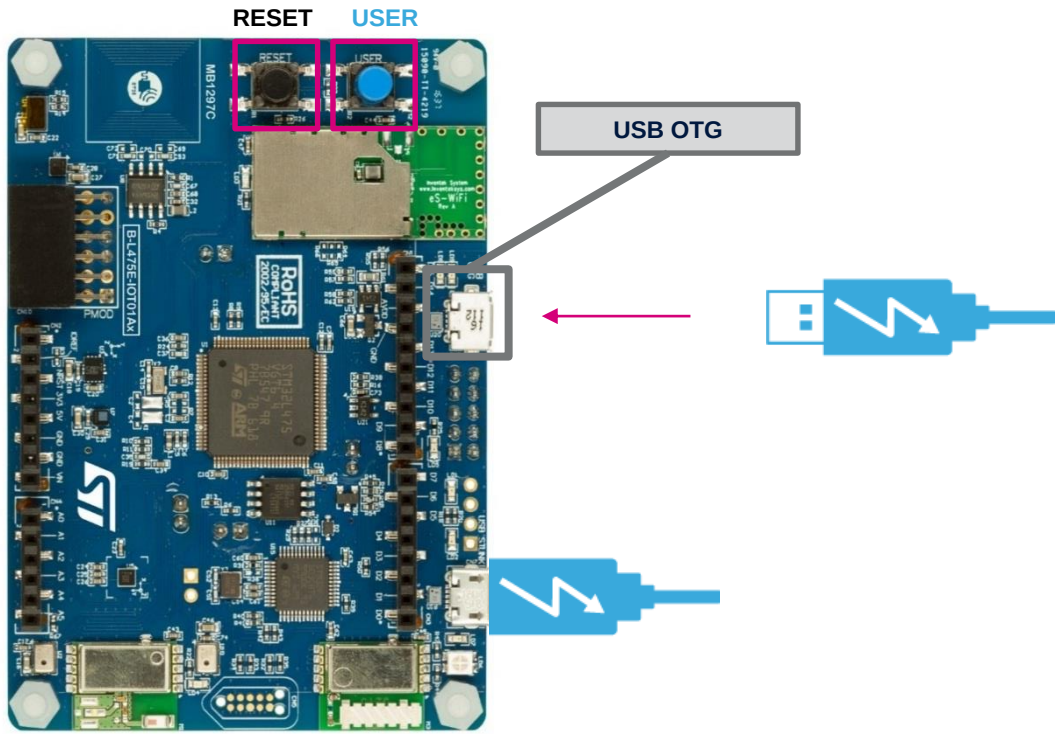
- Data logging feature is available in order to enable users to create their data sets for training their NN models
- Pre-processing python scripts for model re-training are also provided in the package

life.augmented

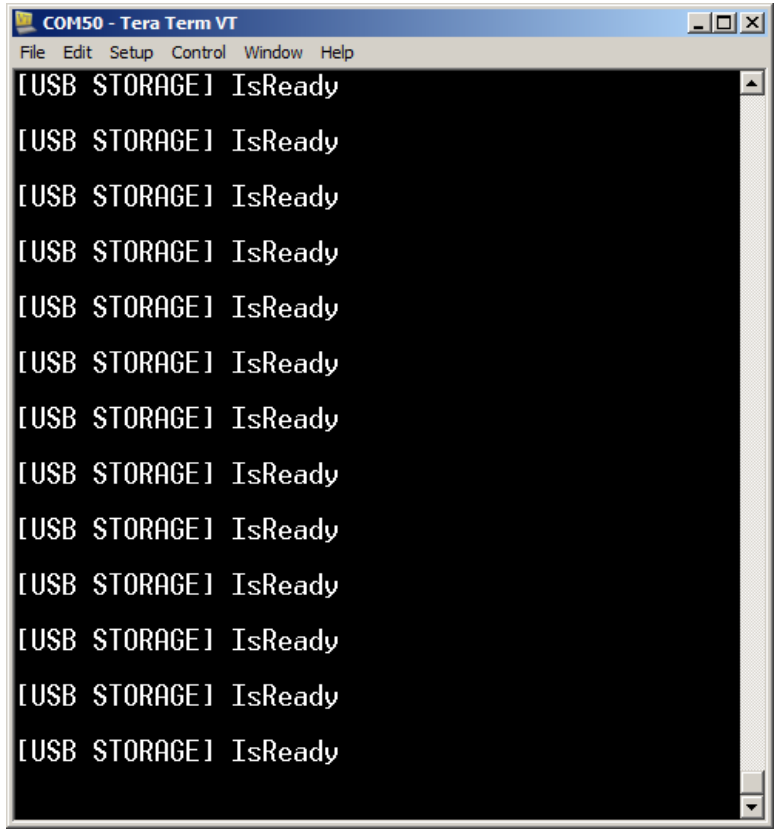
Lab 2: Data Collection

Step 2 – Read .wav file (1/2)

- Connect 2nd USB Cable
- Using terminal, enter usb start command

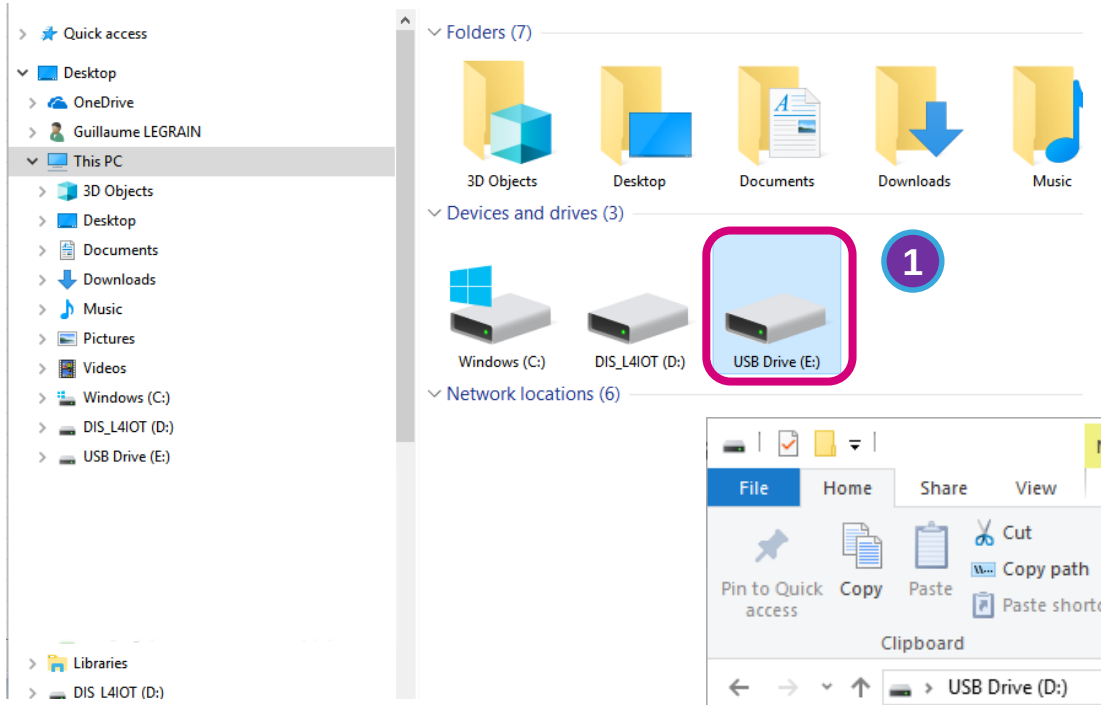


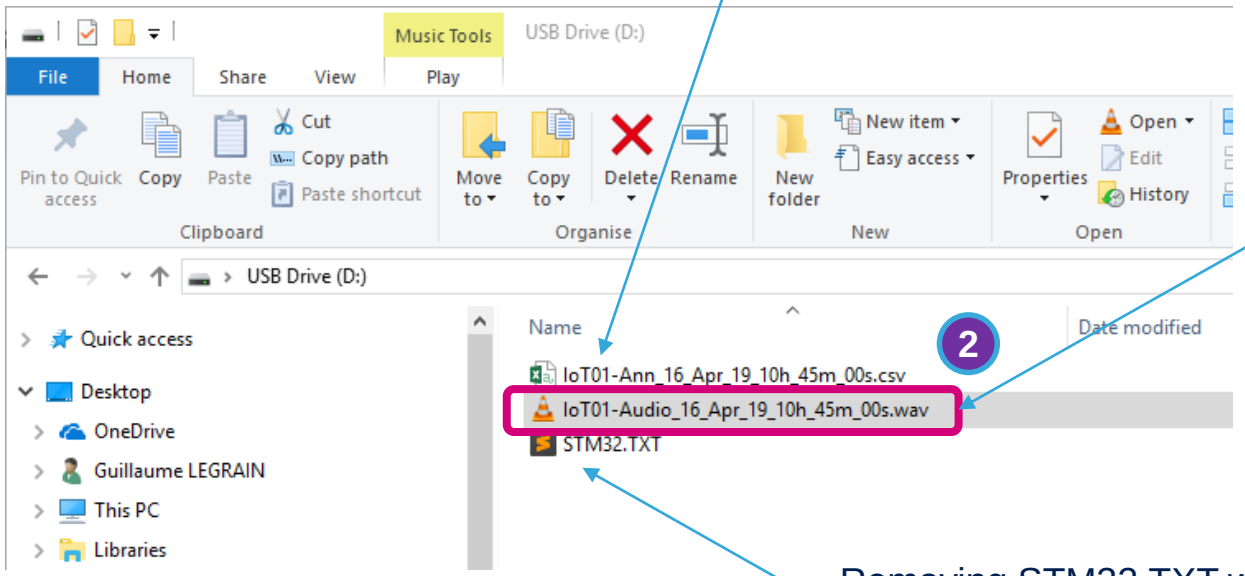

```
$ usb start
```



Lab 2: Data Collection

Step 2 – Read .wav file (2/2)





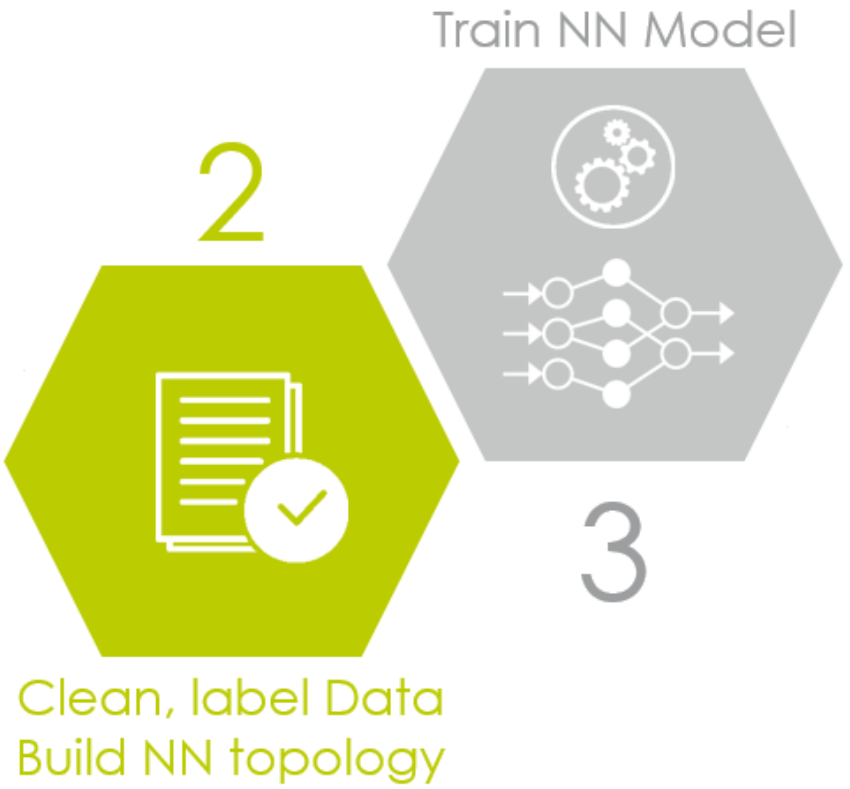
Label timestamps and other sensors data

Audio recording

Removing STM32.TXT will format the drive and erase all QSPI Flash content after MCU reset

life.augmented

Neural Network **Model creation** using Keras



Model Creation

- Gather the Data
  - Download and load data from a public dataset
- Prepare the Data
  - Audio signal framing
- Preprocess the Data
  - Log-Mel Spectrogram feature extraction
- Build the Model
  - Build a Sequential Convolutional Neural Network for audio classification
- Train the Model
  - Train the model using training and validation data
- Evaluate the Model
  - Evaluate model accuracy against new ‘test’ data

python™

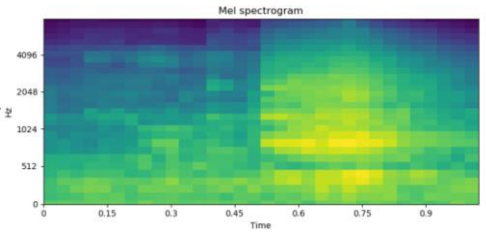
Keras

Model Creation

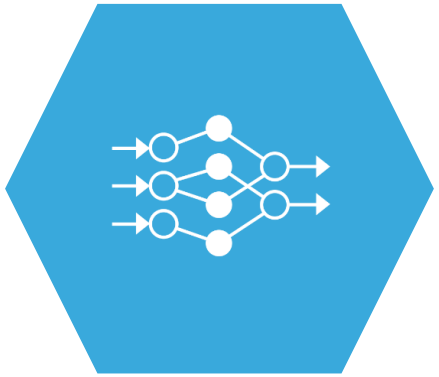
Acoustic Scene Classification pipeline

Input signal sampling

Log-Mel Preprocessing



Convolutional Neural Network

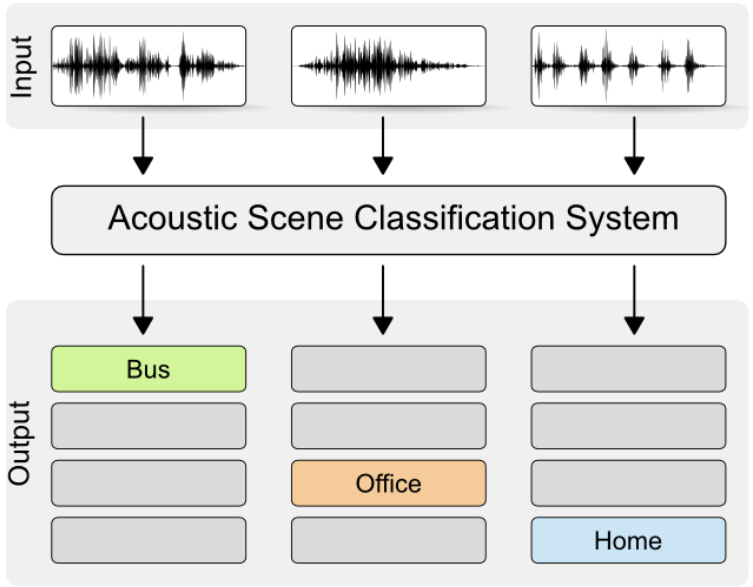


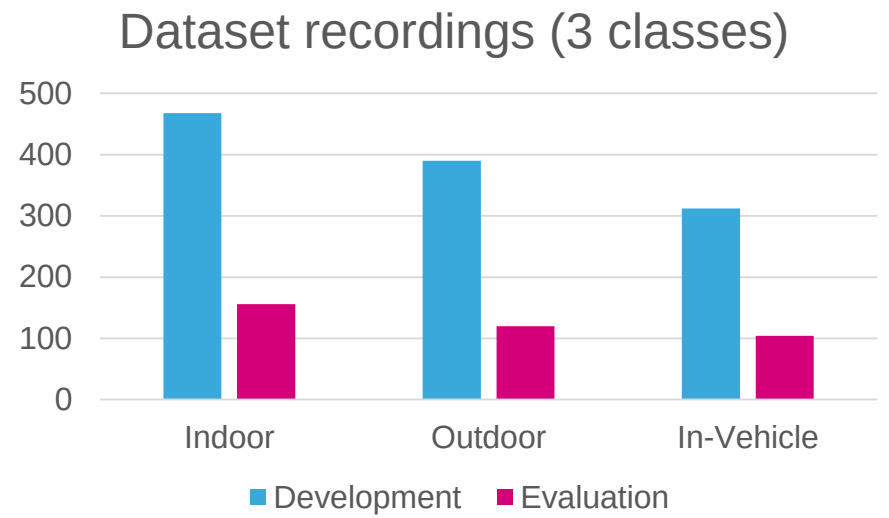
Classification

“Indoor”	**0.92**
“Outdoor”	0.07
“In-Vehicle”	0.01

Model Creation

TUT Acoustic scenes 2016 dataset





- 44.1kHz
- Stereo
- 24-bit integers
- 15 Acoustic scenes (Bus, car, train …)

→

- 16kHz
- Mono
- Single-precision floating-point
- 3 Acoustic scenes (indoor, outdoor, in-vehicle)

life.augmented

Model Creation

Load the *TUT Acoustic scenes 2016* dataset

```
import tutacousticscenes2016

# Download and load the TUT Acoustic scenes 2016 dataset
(x_dev, y_dev), (x_eval, y_eval) = tutacousticscenes2016.load_data()

print('x_dev shape:', x_dev.shape)
print(x_dev.shape[0], 'development samples')
print(x_eval.shape[0], 'evaluation samples')
```

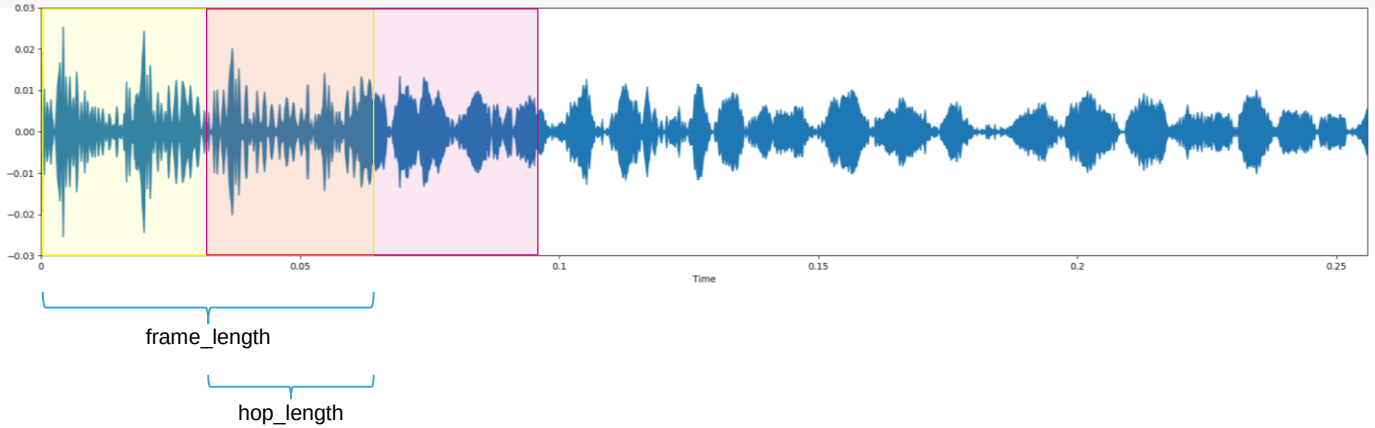
```
Downloading data from https://zenodo.org/record/45739/files/TUT-acoustic-scenes-2016-development.audio.1.zip
1070989312/1070981236 [==============================] - 132s 0us/step
Downloading data from https://zenodo.org/record/45739/files/TUT-acoustic-scenes-2016-development.audio.2.zip
...
Extracting ./datasets/TUT-acoustic-scenes-2016/TUT-acoustic-scenes-2016-development.audio.1.zip
Extracting ./datasets/TUT-acoustic-scenes-2016/TUT-acoustic-scenes-2016-development.audio.2.zip
...


x_dev shape: (1170, 480001)
1170 development samples (1170 30s segments)
390 evaluation samples (390 30s segments)
```

Model Creation

Prepare the data

```
import numpy as np
import librosa.util

# Slice the 30s signal segment into overlapping frames of 64ms long to return 936 frames of 1024 samples
x_dev_framed = []
for i in range(x_dev.shape[0]):
    frames = librosa.util.frame(x_dev[i], frame_length=1024, hop_length=512)
    x_dev_framed.append(np.transpose(frames))

print(len(x_dev_framed))
print(x_dev_framed[0].shape)
```

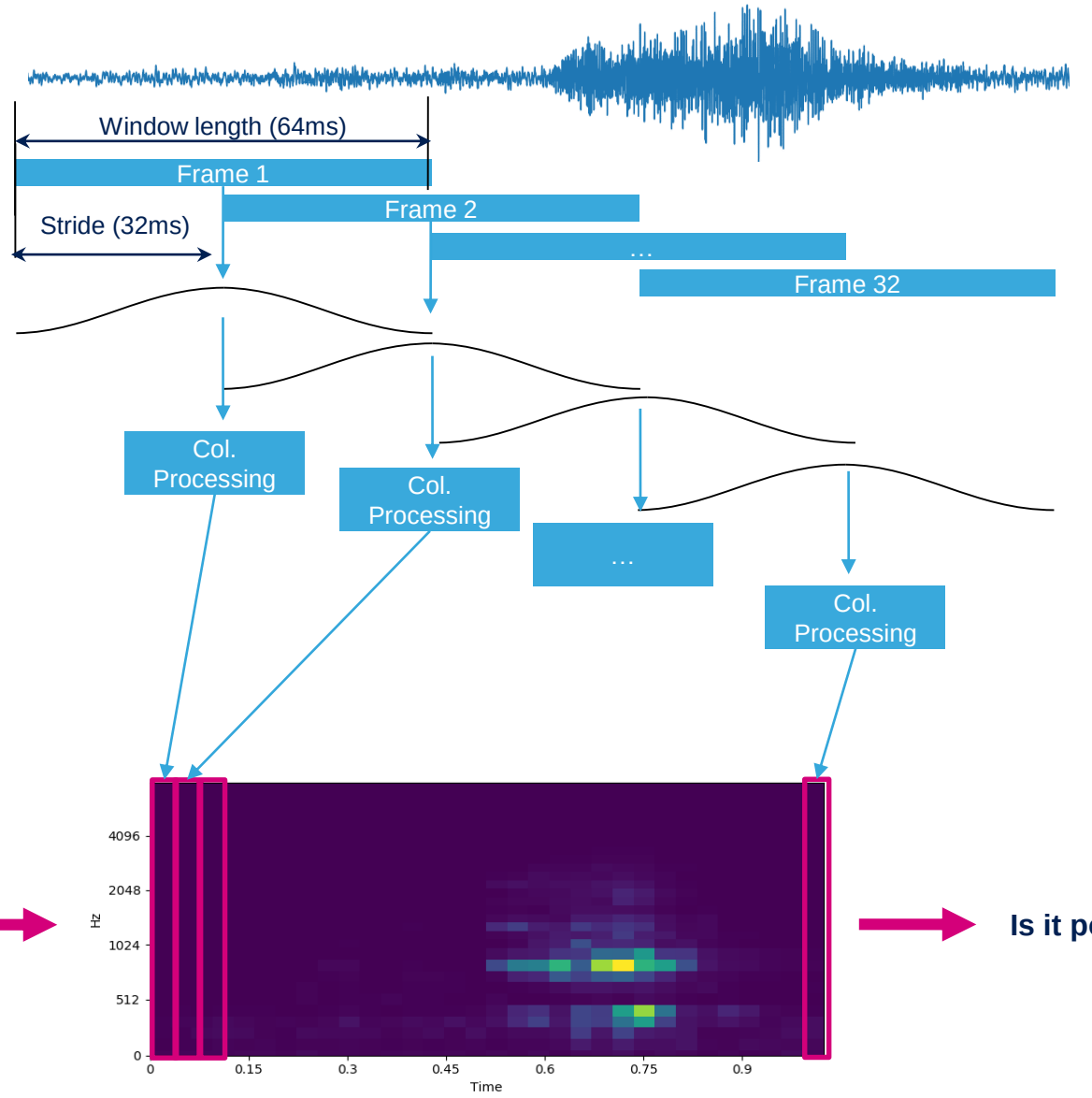
1170
(936, 1024)



life.augmented

Model Creation

Feature Extraction



Picture to be recognized ? → **Is it possible to get more details/features ?**

life.augmented

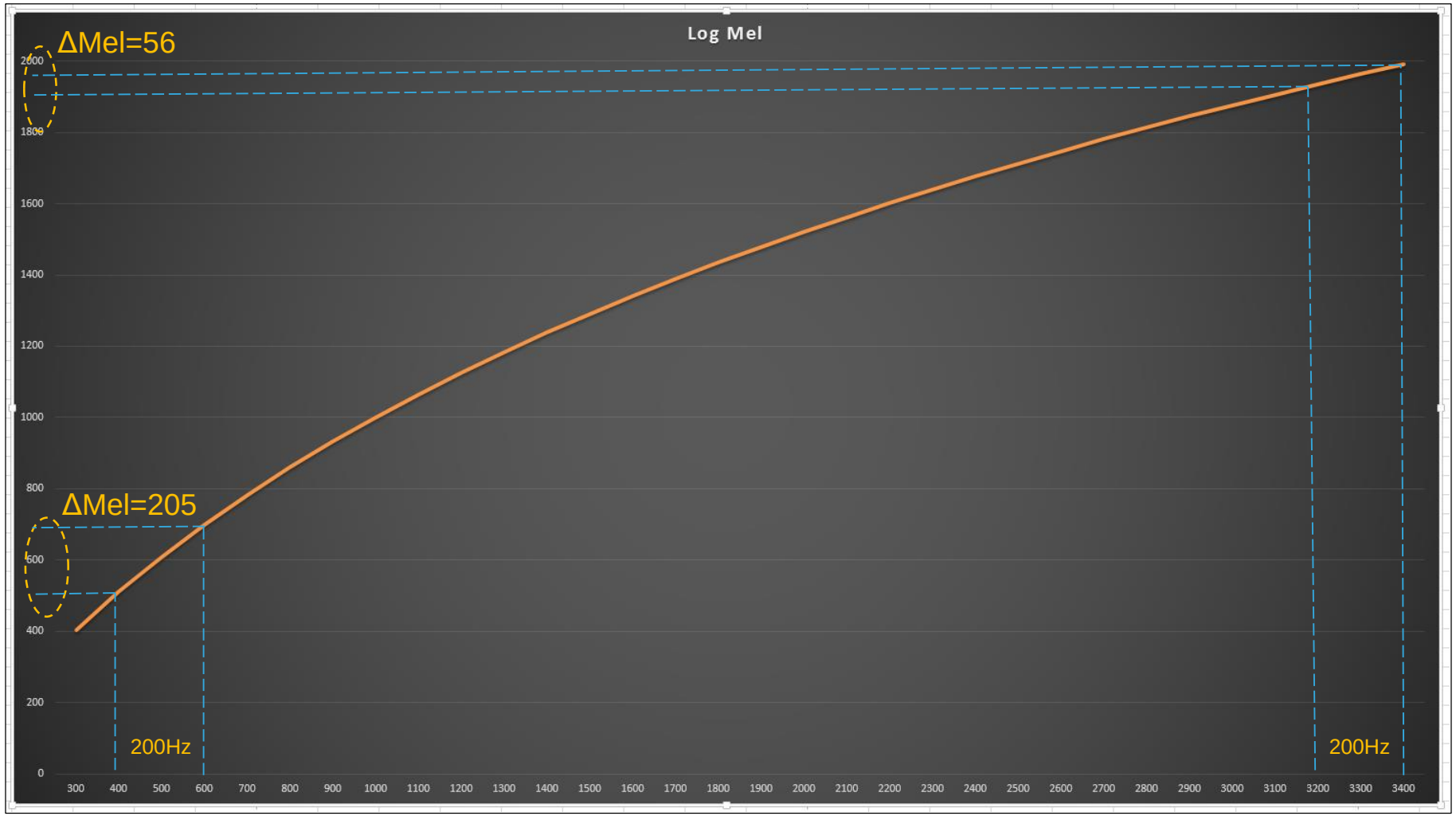
Model Creation

Mel Scale

Humans are much better at discerning small changes in **perceived** pitch at low frequencies than at high frequencies i.e. the perceived ‘distance’ between 300Hz and 500Hz tone seems to us to be bigger than between 3200Hz and 3400Hz.

We can find a relation between perceptual pitch and measured frequency as logarithmic function.

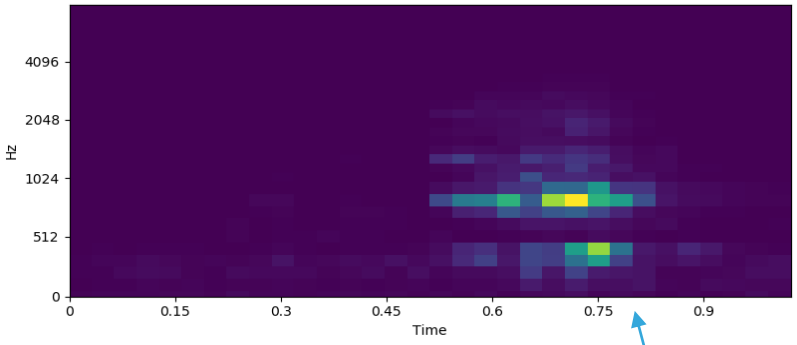
$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$



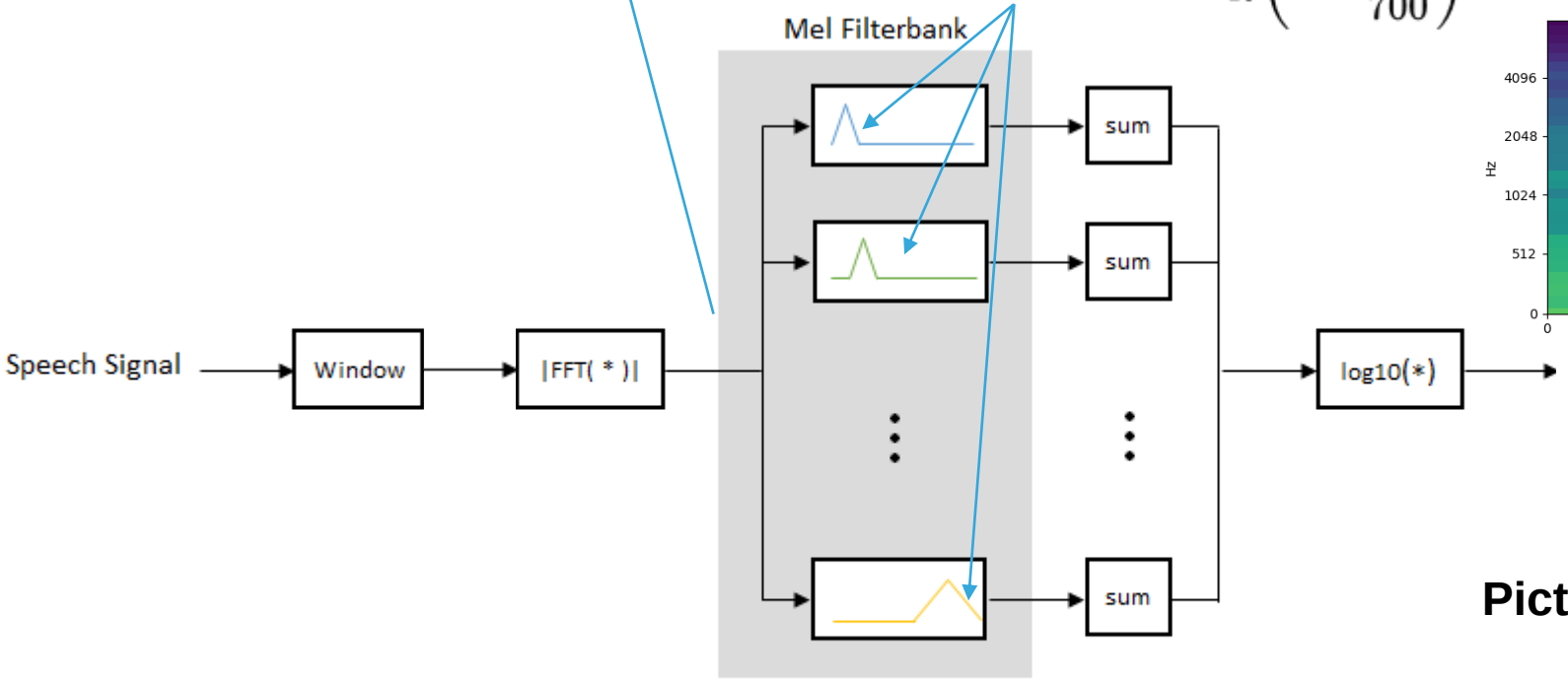
Source: Modified from https://en.wikipedia.org/wiki/Mel_scale

Model Creation

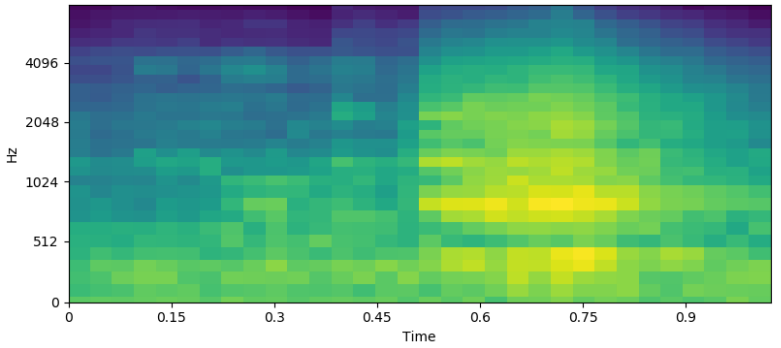
Feature Extraction



$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$



LogMel filtering



Picture ready to be recognized by ANN

Source: Modified from https://www.mathworks.com/help/audio/examples/speaker-identification-using-pitch-and-mfcc.html
https://en.wikipedia.org/wiki/Mel_scale

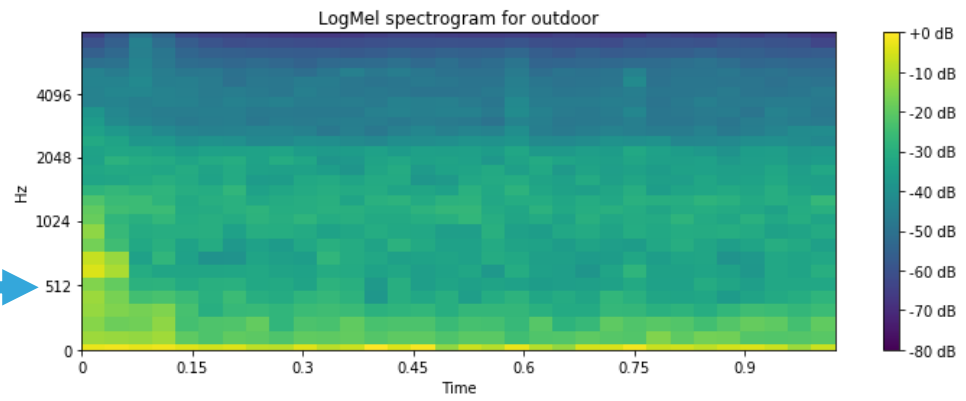
life.augmented

Model Creation

Pre-process the data using Python

```
from LogMelSpectrogram import feature_extraction

def preprocess_data(x_framed, y):
    n_features_per_file = int(x_framed[0].shape[0] / 32) # 29 spectrograms per 30s audio file
    n_files = len(x_framed)
    x_features = np.empty((n_files * n_features_per_file, 30, 32), dtype='float32', order='C')
    y_features = np.empty(n_files * n_features_per_file, dtype='int16')
    for i in range(0, n_files):
        for j in range(0, n_features_per_file):
            frame = x_framed[i][0 + j:32 + j]
            x_features[i * n_features_per_file + j] = feature_extraction(frame)
            y_features[i * n_features_per_file + j] = y[i]

return (x_features, y_features)

x_dev_features, y_dev_features = preprocess_data(x_dev_framed, y_dev)
x_eval_features, y_eval_features = preprocess_data(x_eval_framed
                                                   ))

print(x_dev_features.shape)
print(x_eval_features.shape)
```

(33930, 30, 32) -> 33930 1.024s 30x32 “pictures”
(11310, 30, 32) -> 11310 1.024s 30x32 “pictures”



life.augmented

Model Creation

One-hot encode expected output vectors

```
from keras.utils import to_categorical

# Convert labels to categorical one-hot encoding
y_dev_features_cat = to_categorical(y_dev_features, num_classes=3)
y_eval_features_cat = to_categorical(y_eval_features, num_classes=3)

y_dev_features_cat.shape
```

```
(33930, 3)
```

`to_categorical()` Converts a class vector (integers) to binary class matrix.

```
[1, 1, 2, ..., 0, 0, 0]      -->     [[0., 1., 0.],
                                      [0., 1., 0.],
                                      [0., 0., 1.],
                                      ...,
                                      [1., 0., 0.],
                                      [1., 0., 0.],
                                      [1., 0., 0.]]
```

Model Creation

Standardize features

Standardize features by **removing the mean** and **scaling to unit variance**
The standard score of a sample x is calculated as:

$$z = \frac{x - \mu}{\sigma}$$

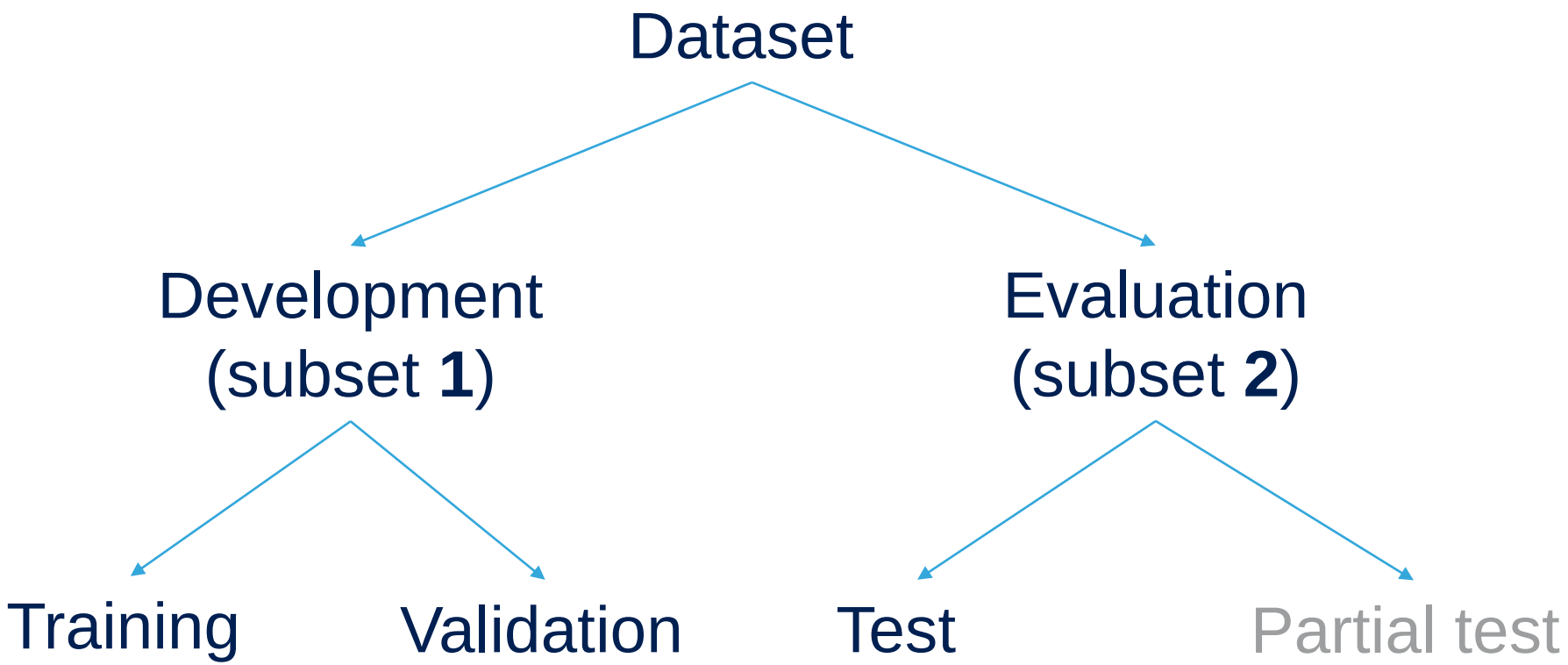
Where:

- μ is the mean of the training samples
- σ is the standard deviation of the training samples.

```
from sklearn import preprocessing

# Flatten features for scaling
x_dev_features_r = x_dev_features.reshape(len(x_dev_features), 30 * 32)
x_eval_features_r = x_eval_features.reshape(len(x_eval_features), 30 * 32)

# Create scaler using only the development dataset
scaler = preprocessing.StandardScaler().fit(x_dev_features_r)

# Apply the same scaler to the development and evaluation set
x_dev_features_s = scaler.transform(x_dev_features_r)
x_eval_features_s = scaler.transform(x_eval_features_r)
```

Model Creation

NN learning dataset



Model Creation

Split *development* set into *train* and *validation* set

When training, we want to **check the accuracy of the model on data it hasn't seen before**. So we wan to **split the development dataset into a training and validation set** to evaluate the loss and other model metrics at the end of each training epoch. The goal is to develop and tune the model using only the development data. The evaluation dataset will only be used for final evaluation as if it was unknown.

```
x_train, x_val, y_train, y_val = train_test_split(x_dev_features_s, y_dev_features_cat, test_size=0.25, random_state=1)

x_test = x_eval_features_s.reshape(len(x_eval_features), 30 * 32)
y_test = y_eval_features_cat
_, x_test_partial, _, y_test_partial = train_test_split(x_test, y_test, test_size=0.01)

print('Training samples:', x_train.shape)
print('Validation samples:', x_val.shape)
print('Test samples:', x_test.shape)
print('Partial Test samples:', x_test_partial.shape)
```

```
Training samples: (25447, 960)
Validation samples: (8483, 960)
Test samples: (11310, 960)
Partial Test samples: (114, 960)
```

Training: ‘lectures’

Validation: ‘test at the end of the period’

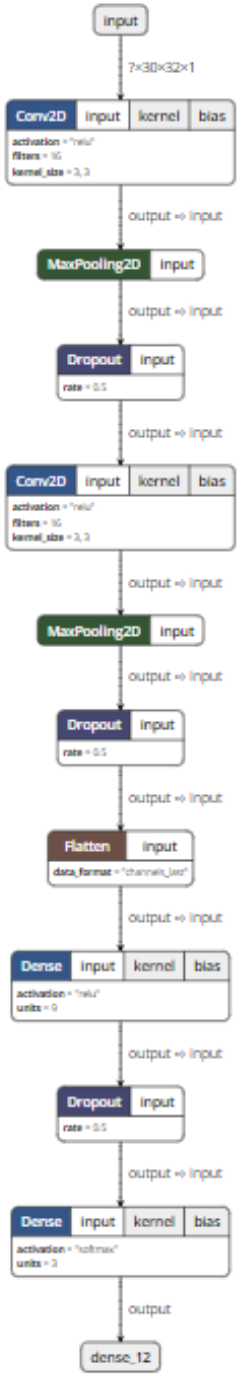
Test: ‘final exam’

Smaller test set for ‘validation on target’

life.augmented

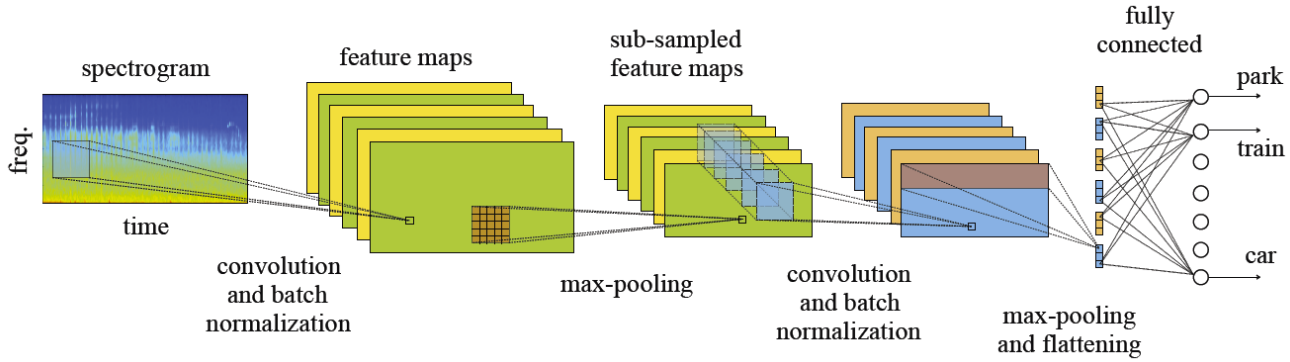
Model Creation

Build the model



NN model:

- Sampling rate: 16KHz
- 1 sec Spectrogram Size: 30x32
- 1° Conv Layer: 16x3x3
- Maxpoling: 2x2
- 2° Conv Layer: 16x3x3
- MaxPooling: 2x2
- 1° Dense Layer: 9 units
- 2° Dense Layer: 3 units
- SoftMax and Argmax(Avg()))

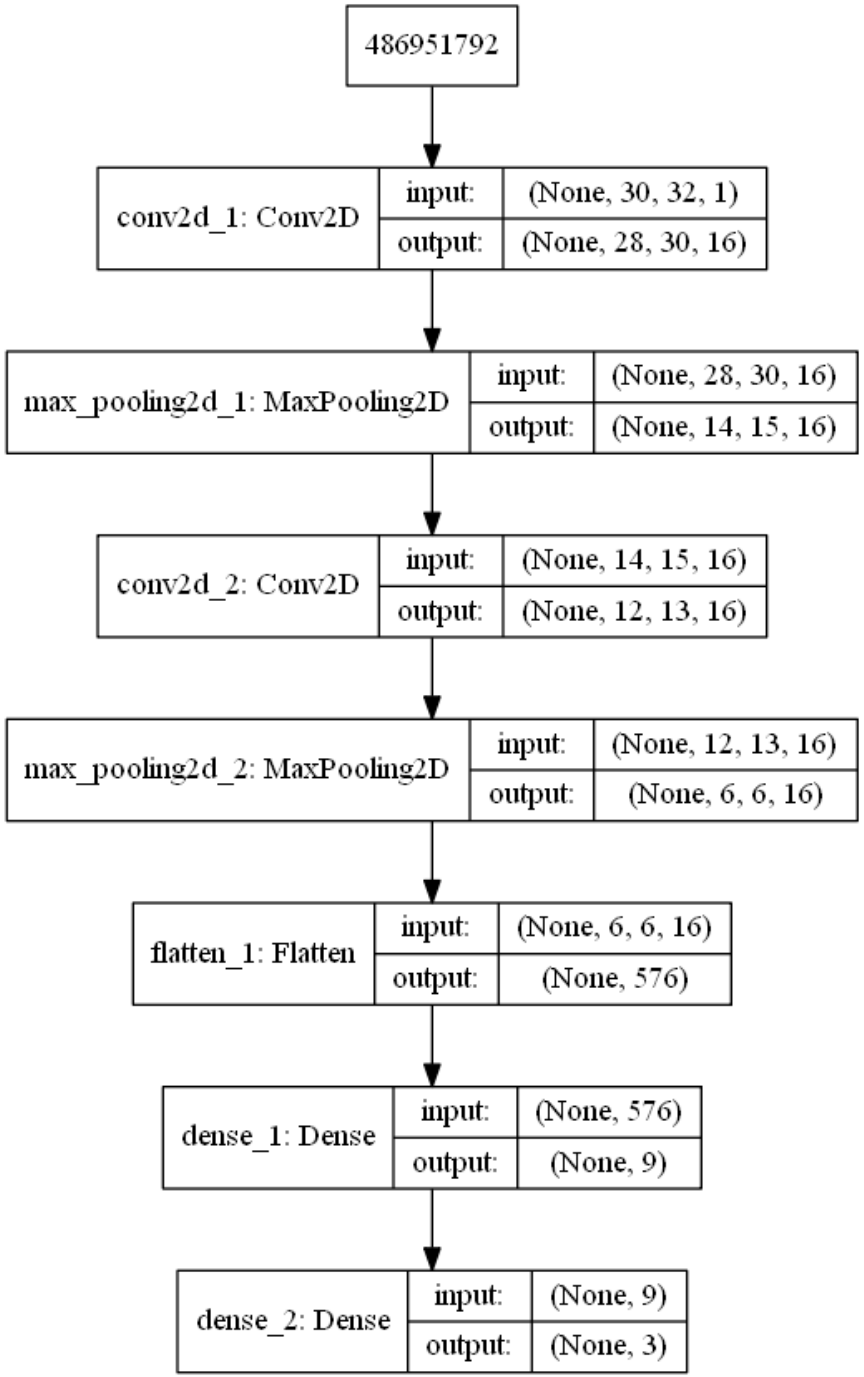



```
model = models.Sequential()
model.add(layers.Conv2D(16, (3, 3), activation='relu', input_shape=(30, 32, 1),
data_format='channels_last'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(16, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(9, activation='relu'))
model.add(layers.Dense(3, activation='softmax'))
```

Based on Ref. DCASE 15 CL Model
Valenti, M., Diment, A., Parascandolo, G., Squartini, S., & Virtanen, T. (2016). DCASE 2016 acoustic scene classification using convolutional neural networks. In *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE 2016), Budapest, Hungary*

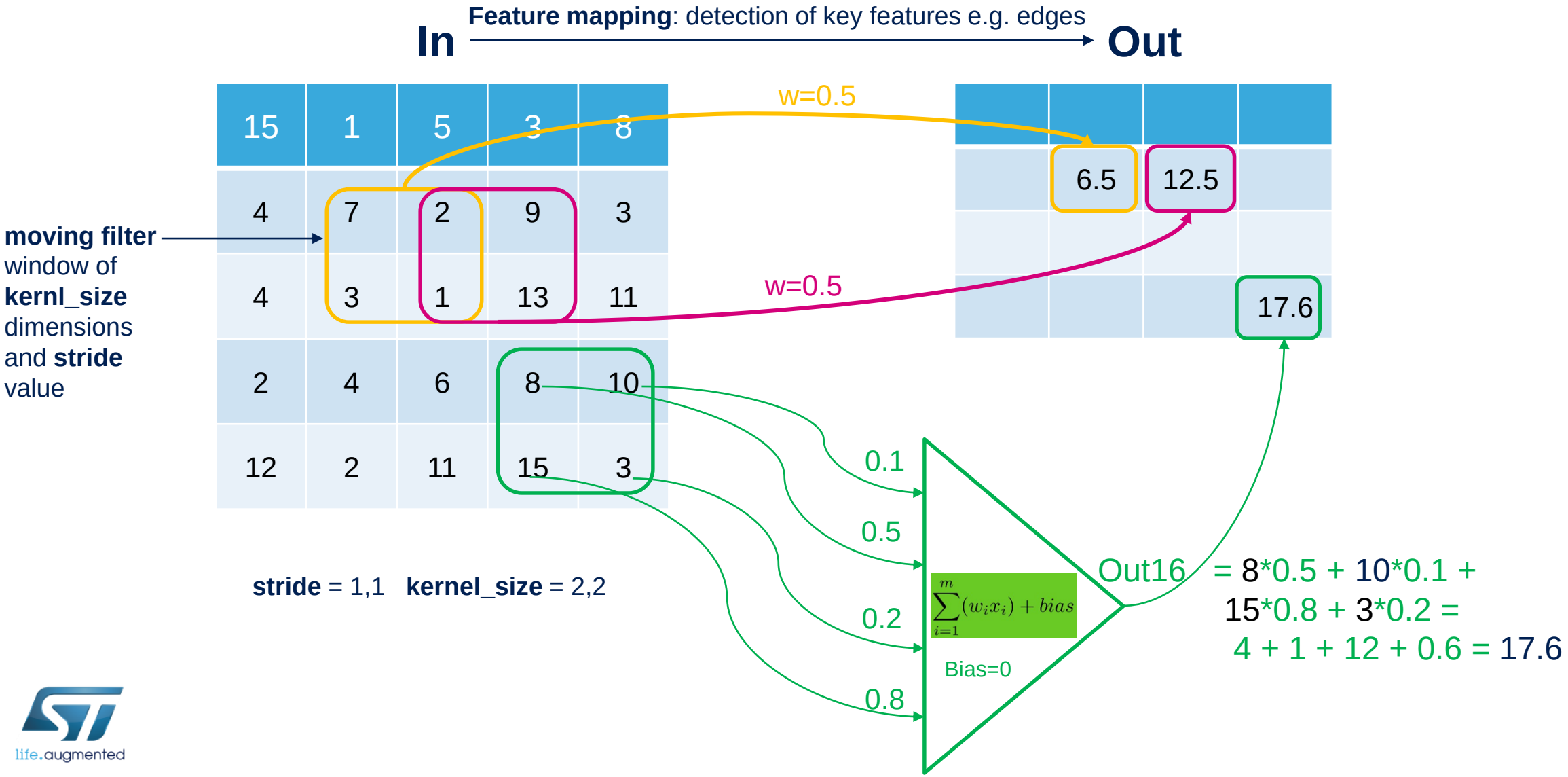
Model Creation

Analyze the model plot



Model Creation

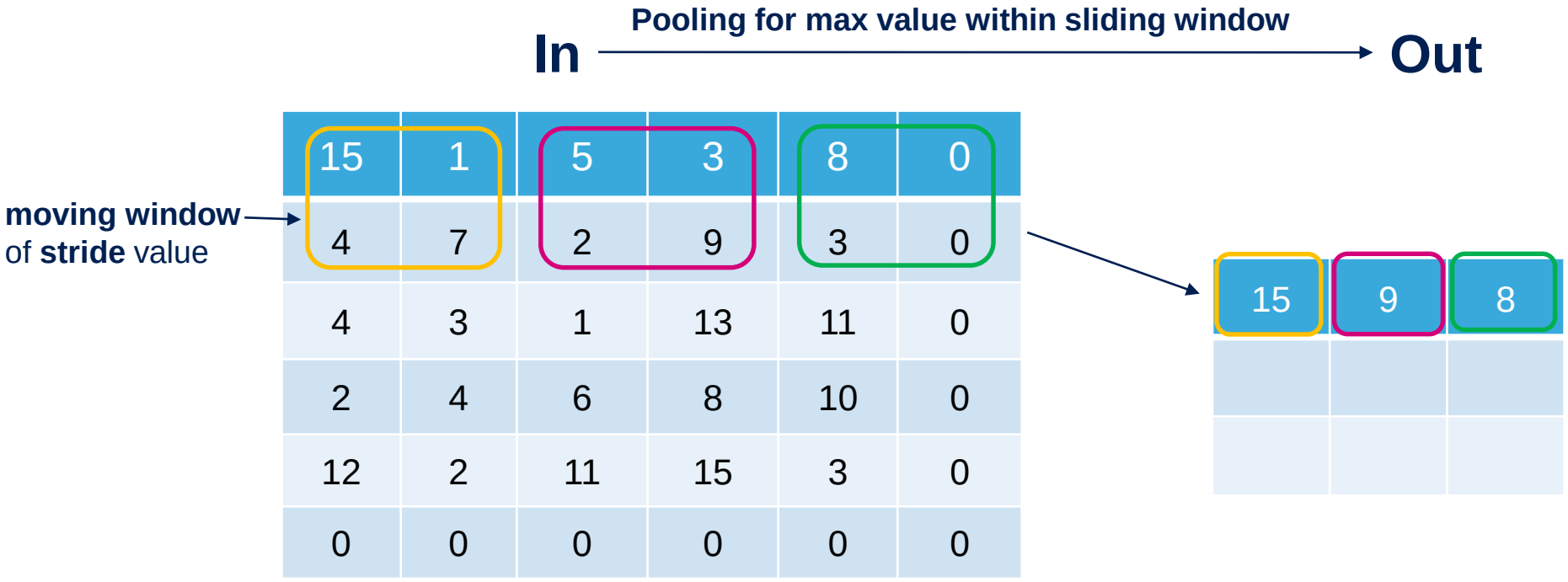
Convolutional layer: let's build ANN based filter



life.augmented

Model Creation

Pooling layer: generalize information

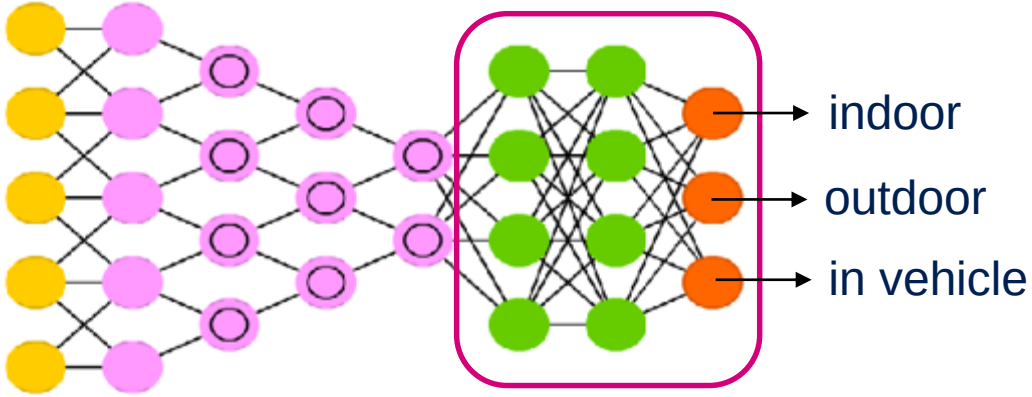


Goals:

- Reduce the amount of information
- Filter the convolutional layer highest probability inferences (outputs)

Model Creation

Fully connected layer: classification of the objects



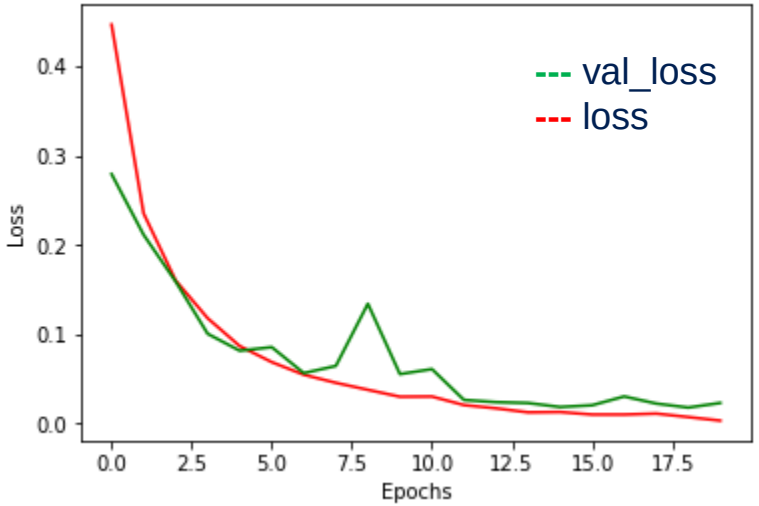
The purpose of fully connected layer is to make classification regarding the objects detected by trained convolutional and max pooling layers

Model Creation

Train the model

```
sgd = optimizers.SGD(lr=0.001, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(optimizer=sgd, loss='categorical_crossentropy', metrics=['acc'])

# Train the model
history = model.fit(x_train_r, y_train, validation_data=(x_val_r, y_val), batch_size=10, epochs=20, verbose=2)
```

```
Train on 25447 samples, validate on 8483 samples
Epoch 1/20
 - 12s - loss: 0.4469 - acc: 0.8315 - val_loss: 0.2793 - val_acc:
0.8892
Epoch 2/20
 - 9s - loss: 0.2353 - acc: 0.9114 - val_loss: 0.2113 - val_acc: 0.9175
...
Epoch 20/20
 - 9s - loss: 0.0030 - acc: 0.9996 - val_loss: 0.0229 - val_acc: 0.9908
```



Model Creation

The **Loss Function** is a function that maps values of one or more variables i.e. NN weights & biases, onto a real number intuitively representing some "**cost**" associated with those values. The goal is to minimize the loss function.

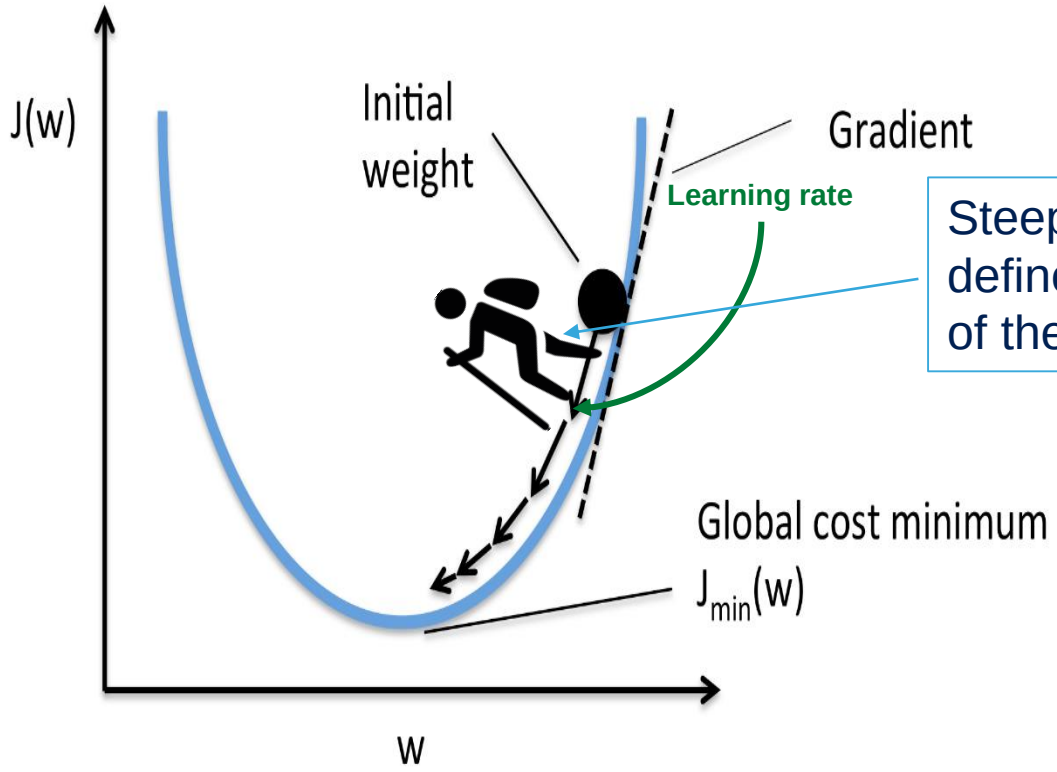
Cost function

Predicted output

Actual output

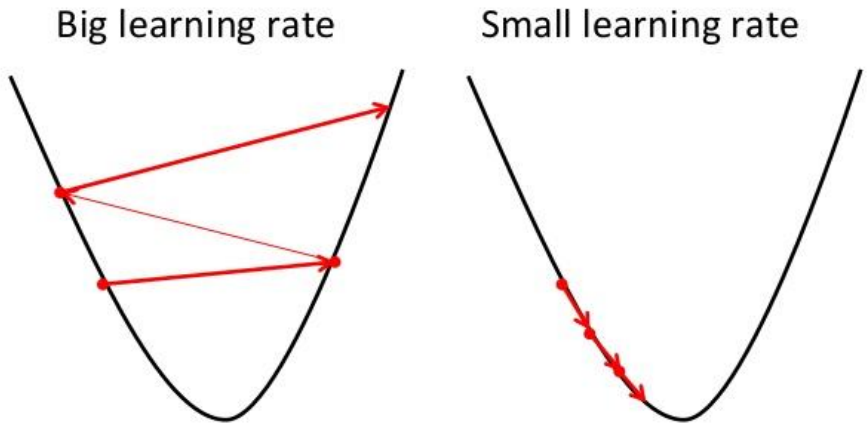
Total no of samples

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$$



Steepest descent defined by negative of the gradient

Gradient Descent



Learning rate compromise is needed:

- Big rate may never converge,
- Small rate will converge but needs a lot of steps.

Model Creation

Evaluate the model - Accuracy

```
loss_and_metrics = model.evaluate(x_test_r, y_test)

print('Test loss:', loss_and_metrics[0])
print('Test accuracy:', loss_and_metrics[1])
```

```
11310/11310 [==============================] - 1s
48us/step
Test loss: 0.7721646421850415
Test accuracy: 0.8928381962864721
```

Model Creation

Evaluate the model - Confusion matrix

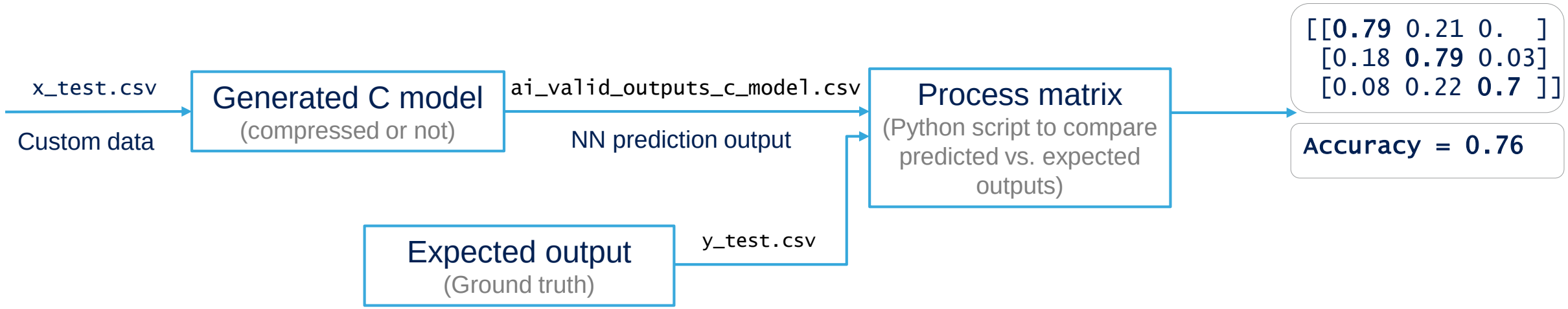
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

y_pred = model.predict(x_val_r)

y_pred_class_nb = np.argmax(y_pred, axis=1)
y_true_class_nb = np.argmax(y_val, axis=1)

matrix = confusion_matrix(y_true_class_nb, y_pred_class_nb, labels=[0,1,2])
accuracy = accuracy_score(y_true_class_nb, y_pred_class_nb)

# (optional) normalize to get values in %
matrix = matrix.astype('float') / matrix.sum(axis=1)[:, np.newaxis]

np.set_printoptions(precision=2)
print(matrix)
print("Accuracy = " + str(accuracy))
```

```
[[0.99 0.   0.01]
 [0.01 0.98 0.01]
 [0.   0.   1.  ]]
Accuracy =
0.990805139691147
```

Model creation

Confusion matrix **accuracy** measurement

- Feed the NN inputs with the same test dataset that was used for model evaluation in Python. *i.e.* **x_test.csv**
- Run “Validate” to get NN predicted outputs by the generated C model (compressed or not) *i.e.* **ai_valid_outputs_c_model.csv**
- Compare predicted outputs against expected outputs (predicted class against actual class). Comparison can be done using a Python script to create a confusion matrix and get a classification accuracy measurement using actual data (vs. random data) (c.f. example script)



Network Training

Save the model and train, val, test data

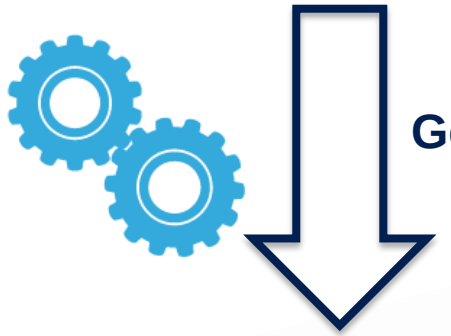
```
# Save the model into an HDF5 file ‘model.h5’
model.save('model.h5')

# save features to csv files in a format X-CUBE-AI can understand, that is, for each tensor, values are in a flattened vector.
np.savetxt('x_train.csv', x_train.reshape(len(x_train), 30 * 32), delimiter=",")
np.savetxt('y_train.csv', y_train, delimiter=",")

np.savetxt('x_val.csv', x_val.reshape(len(x_val), 30 * 32), delimiter=",")
np.savetxt('y_val.csv', y_val, delimiter=",")

np.savetxt('x_test.csv', x_test.reshape(len(x_test), 30 * 32), delimiter=",")
np.savetxt('y_test.csv', y_test, delimiter=",")

np.savetxt('x_test_partial.csv', x_test_partial.reshape(len(x_test_partial), 30 * 32), delimiter=",")
np.savetxt('y_test_partial.csv', y_test_partial, delimiter=",")
```

STM32CubeMX, what is it ?

Key learning

- Overview of the initialization C code generator tool

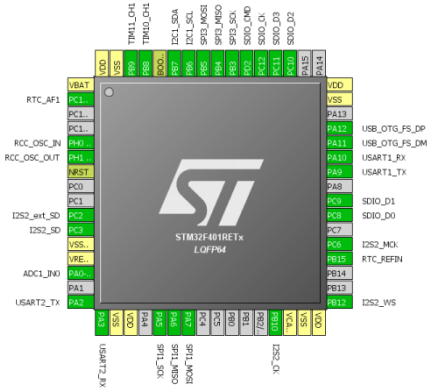
STM32CubeMX



Generates Initialization C Code based on user choices !

life.augmented

STM32CubeMX



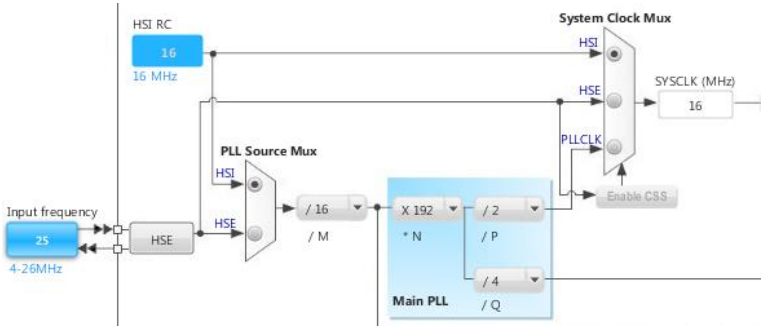
Pinout Configurator

Peripherals & Middleware Configurator

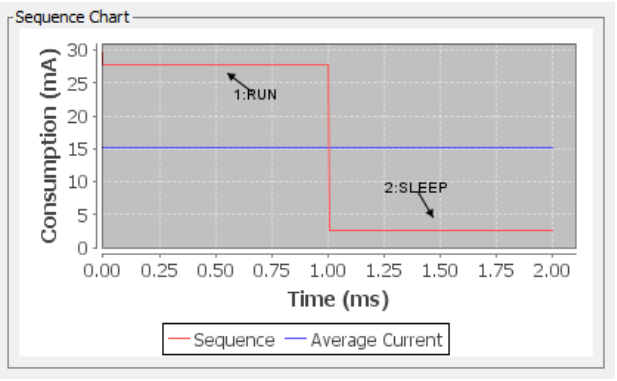
Basic Parameters	
Baud Rate	115200 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
Advanced Parameters	
Data Direction	Receive and Transmit
Over Sampling	16 Samples

Baud Rate
BaudRate must be between **110 Bits/s** and **10.5 MBits/s**.

Clock Tree configurator

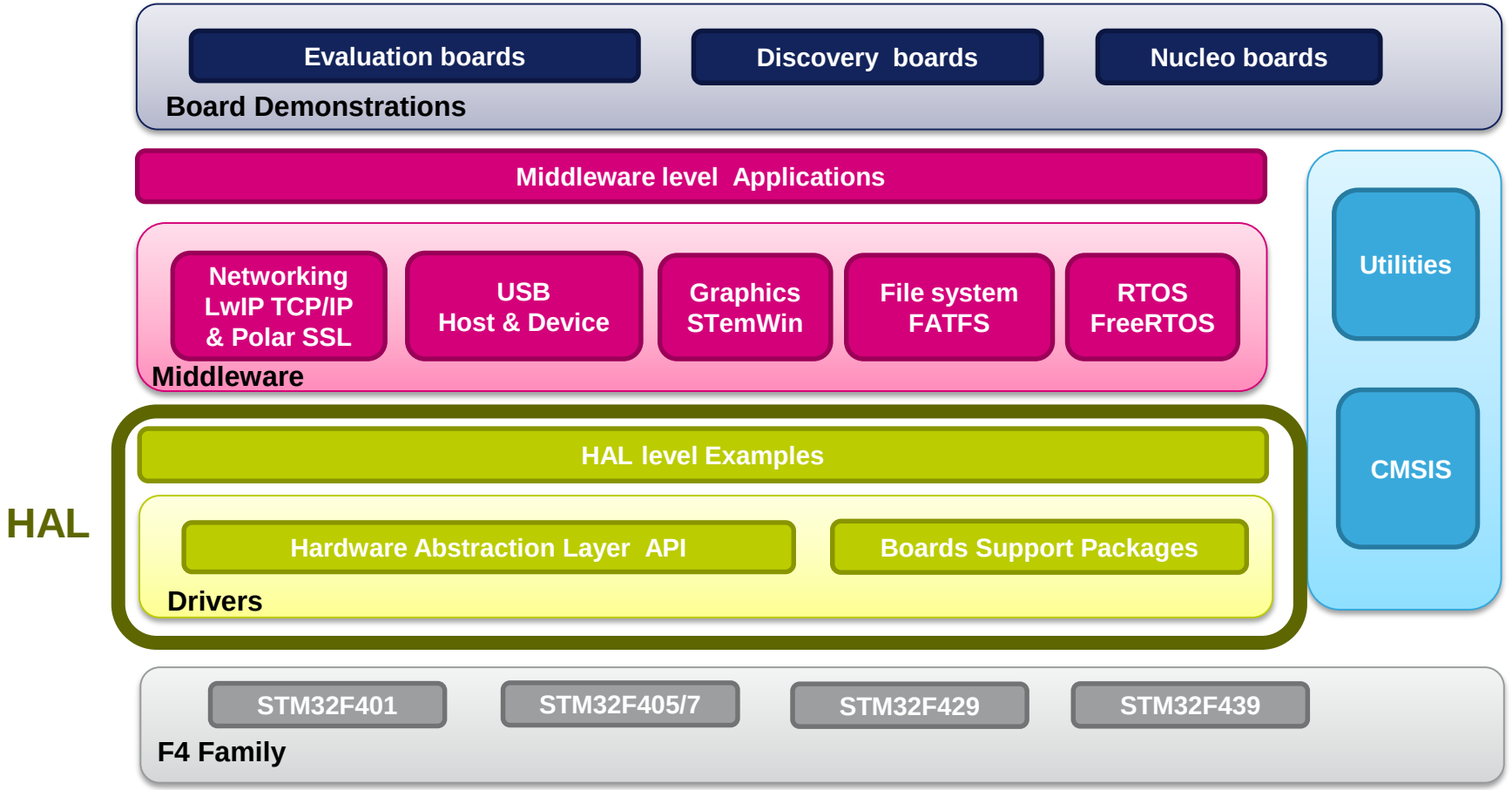


Power Consumption Configurator



life.augmented

HAL general concepts
The HAL in STM32Cube FW package



HAL general concepts

Introduction to HAL

- The STM32Cube Hardware abstraction layer (HAL) is the replacement for the standard peripheral library
- The main objectives of the HAL is to offer
  - **User friendly APIs** that **hide the HW complexity** and focus on the **functionality**
  - **Portable APIs** that allowing **easy migration** of user application across different product families
- All HAL drivers follow a strict C coding rules and were tested using CodeSonar C code static analysis tool from GrammaTech
- HAL documentation is provided as a PDF manual based on Doxygen extracts
- Documentation is in Drivers\STM32XXXX_HAL_Driver\STM32XXXX_HAL_Driver_UM.chm

See STM32CubeMX & HAL library online training

https://www.st.com/content/st_com/en/support/learning/stm32-education/stm32-moocs/stm32cubemx-and-cubeHhal-basics.html

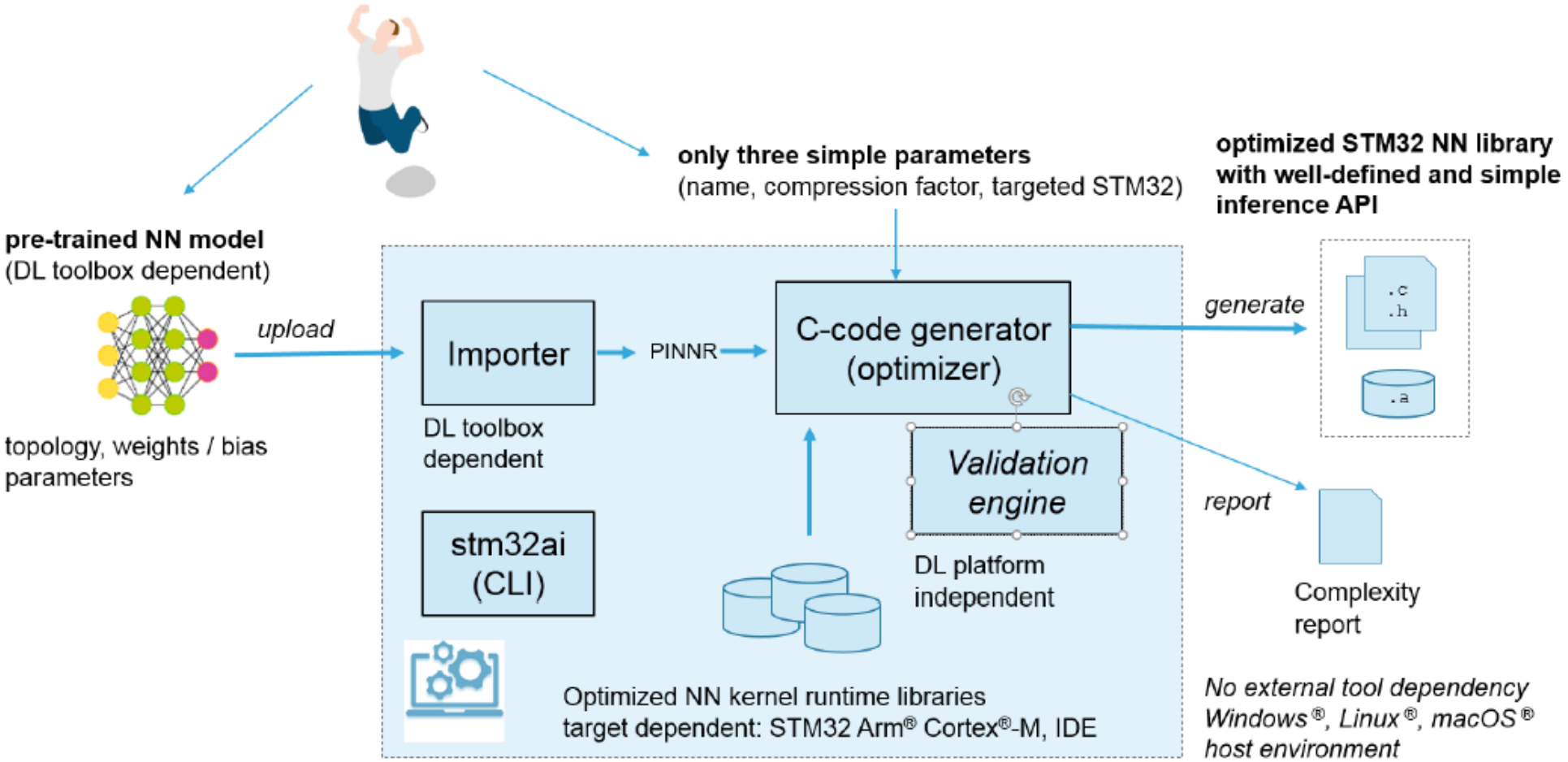
Lab 3: X-CUBE-AI

How to configure and generate source code using **STM32CubeMX** and **X-CUBE-AI**

Key learning

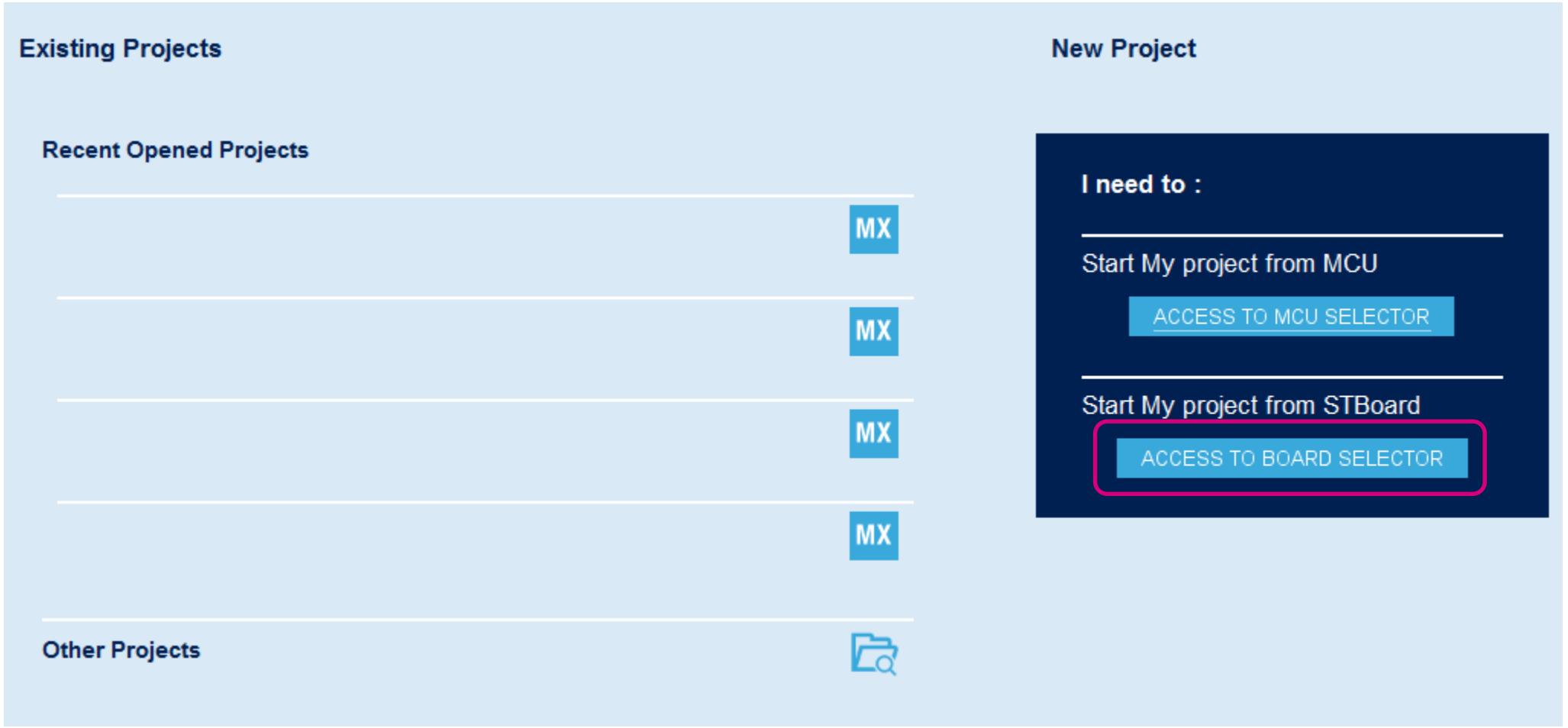
- Configuration of X-CUBE-AI and related AI applications using STM32CubeMX GUI
- Load of neural network model and generation of C code based software project
- Explanation of generated project structure
- Validation of C code based NN using relative error approach on target board

X-CUBE-AI: NN AI expansion package for STM32CubeMX



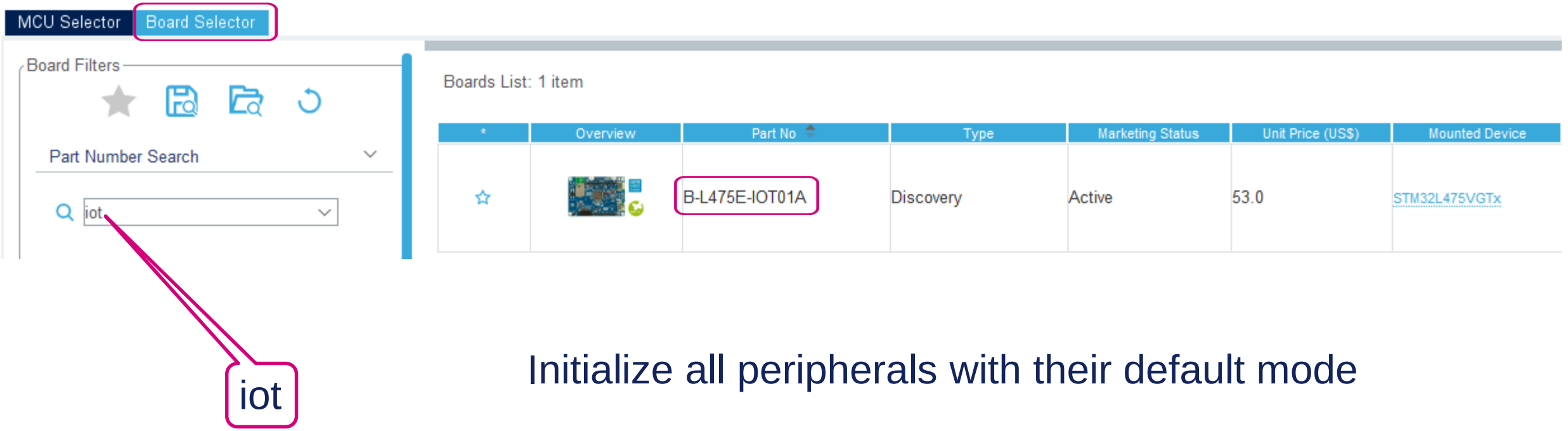
Lab 3: X-CUBE-AI

Open STM32CubeMX and start project from STBoard

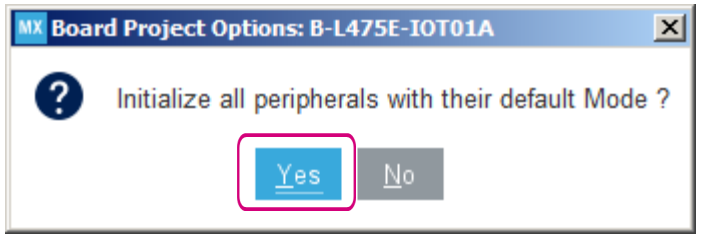


Lab 3: X-CUBE-AI

Select B-L475-IOT01A board



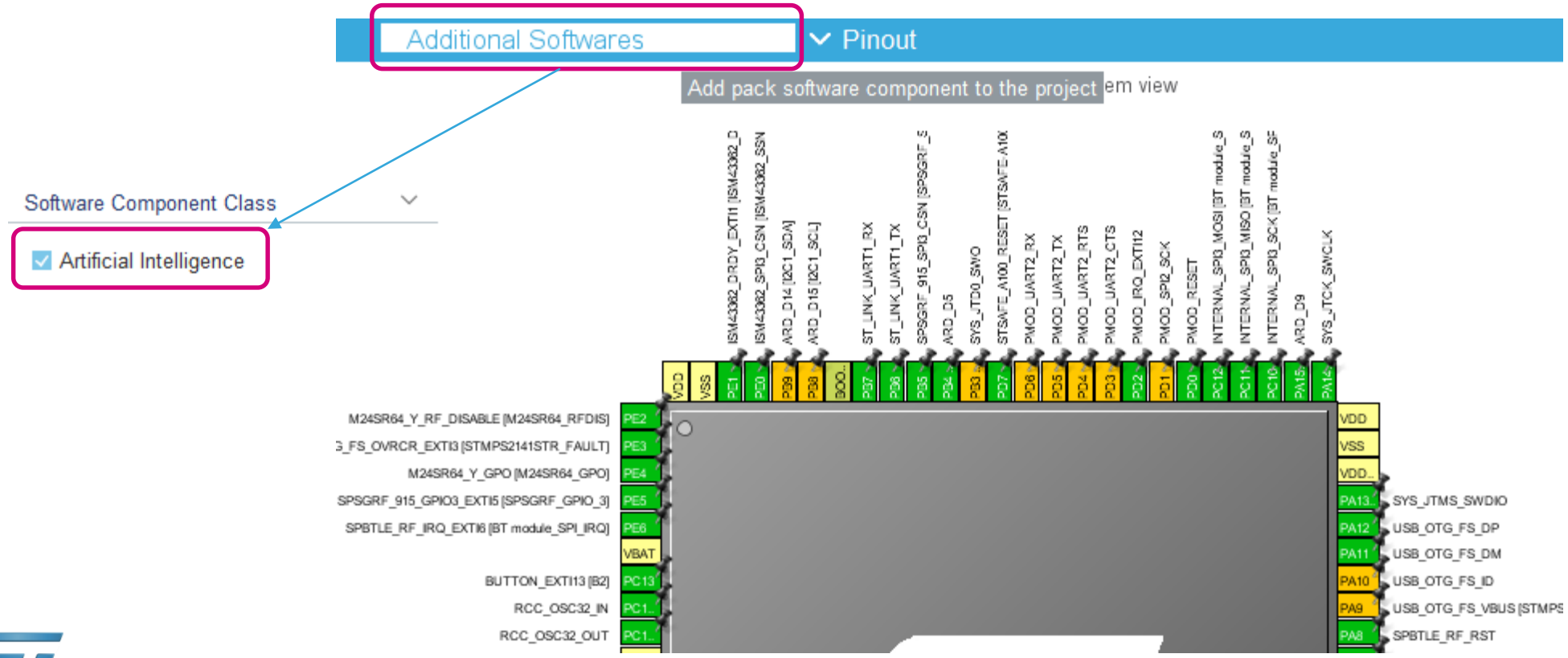
Initialize all peripherals with their default mode



life.augmented

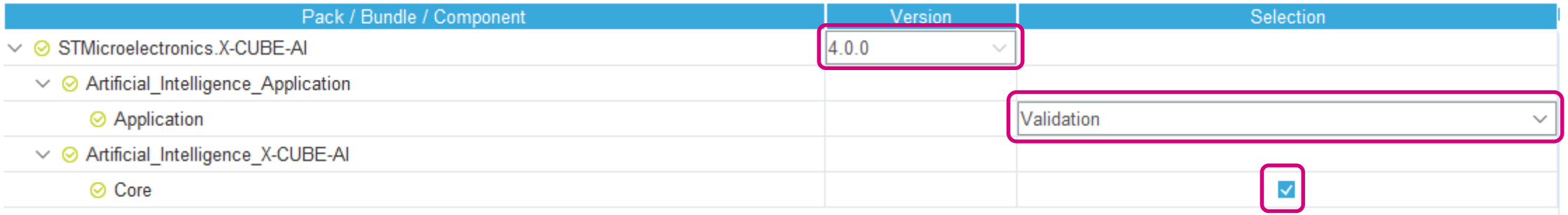
Lab 3: X-CUBE-AI

Press **Additional Software** button within **Pinout & Configuration** tab and check **Artificial Intelligence**



Lab 3: X-CUBE-AI

Check X-CUBE-AI 4.0.0 **Core/** item,
Select X-CUBE-AI application as **Validation**,



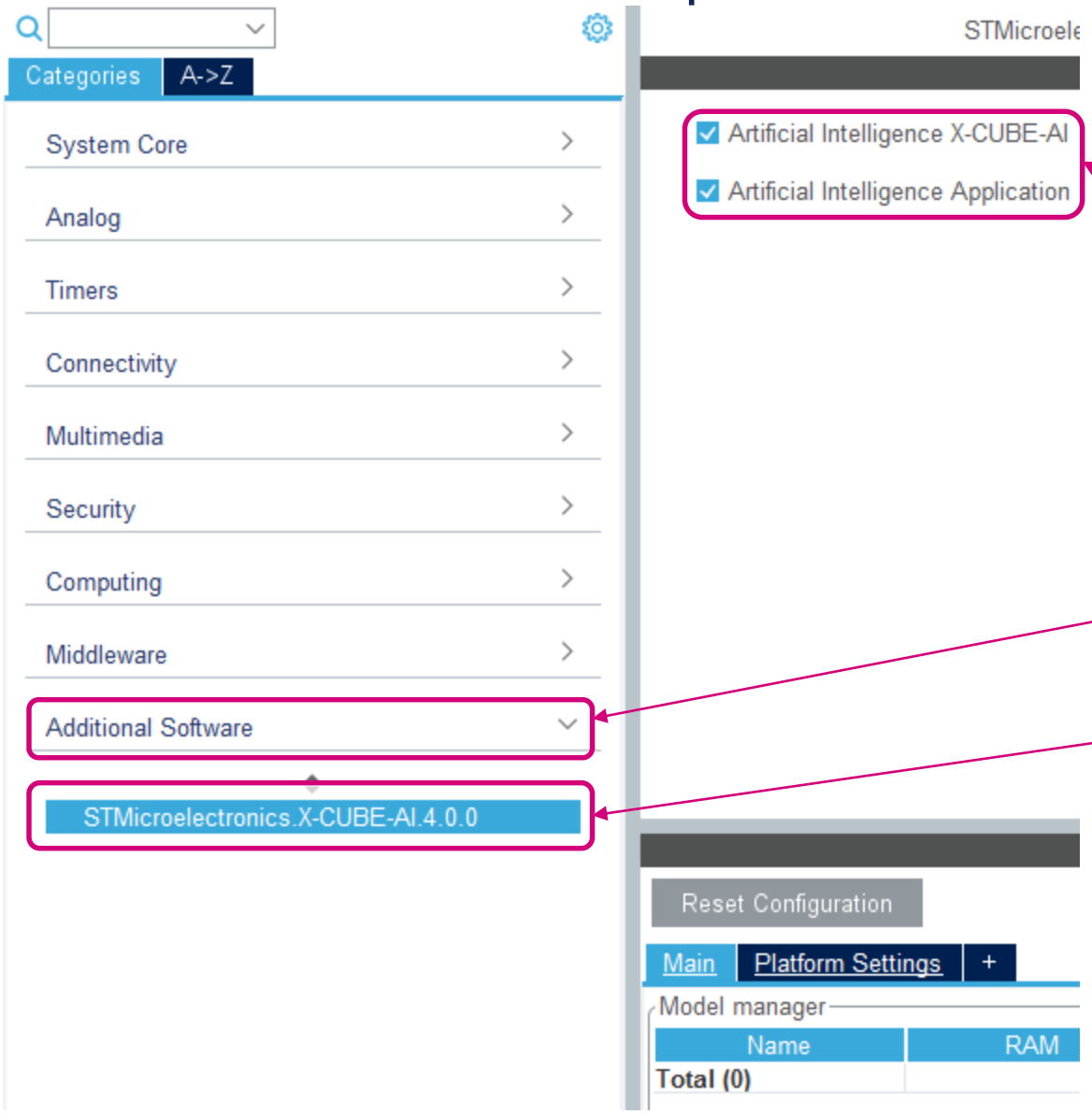
Press **Ok** button

AI applications

- System Performance
  - Complete application project running on the STM32 MCU allowing the accurate measurement of the NN inference CPU load and memory usage. Results are monitored using a Serial Terminal (*e.g.* Tera Term).
- Validation
  - Complete application that validates incrementally the results returned by the NN, stimulated by either random or user test data, on both desktop PC and STM32 Arm® Cortex®-M-based MCU embedded environment. To be used with the X-CUBE-AI validate tool.
- Application Template
  - Empty template project allowing the building of an application including multi-network support.

Lab 3: X-CUBE-AI

Activate X-CUBE-AI sw pack within **Pinout & Configuration** tab



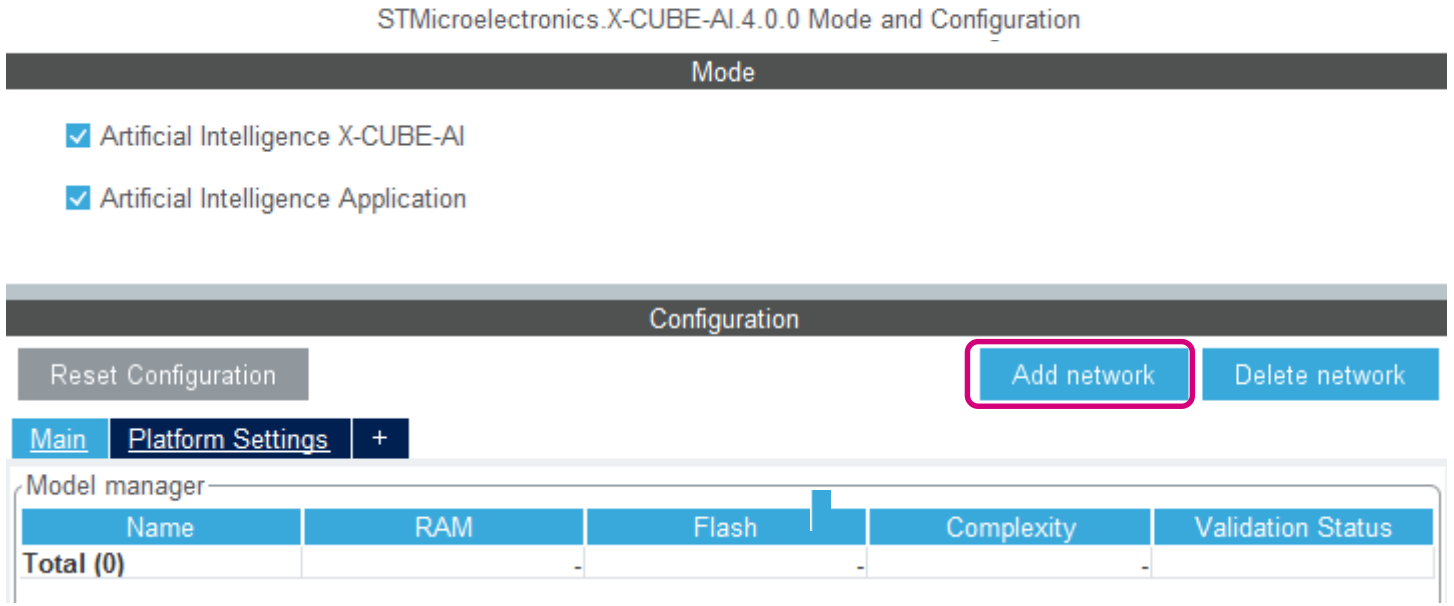
Check both options

Select **Additional Software**,

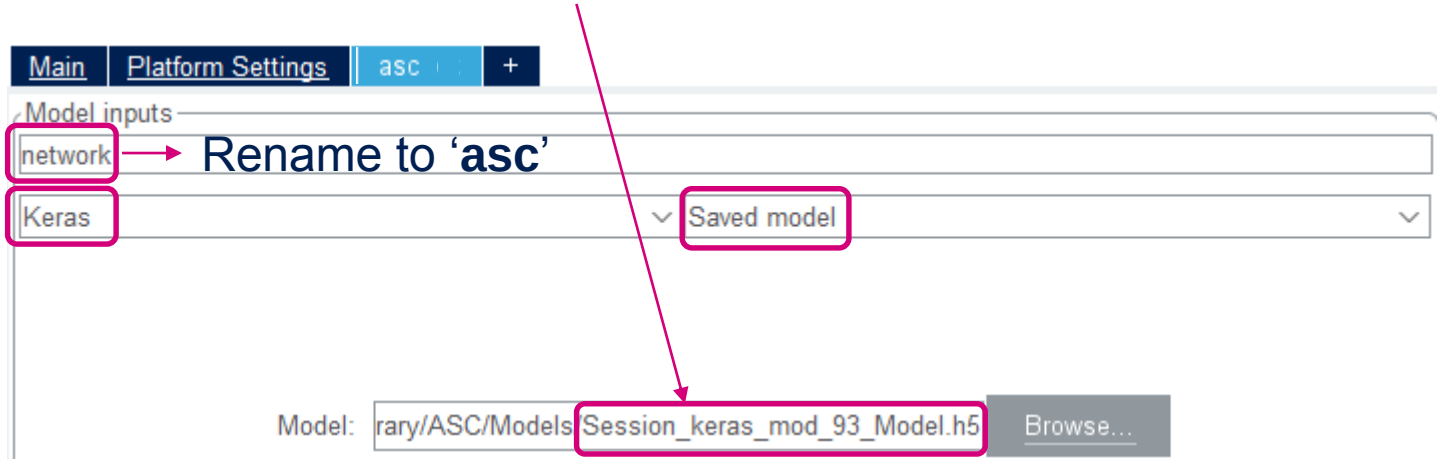
Select **STMicroelectronics X-CUBE-AI.4.0.0**,

Lab 3: X-CUBE-AI

Add NN model (Keras)



C:\AI\FP-AI-SENSING1\Utilities\AI_Resources\models**Session_keras_mod_93_Model.h5**



life.augmented

Lab 3: X-CUBE-AI

See graph optionally

Analyze network in terms of MCU resources

Press **Analyze** button

Compression: None

Validation inputs: Random numbers

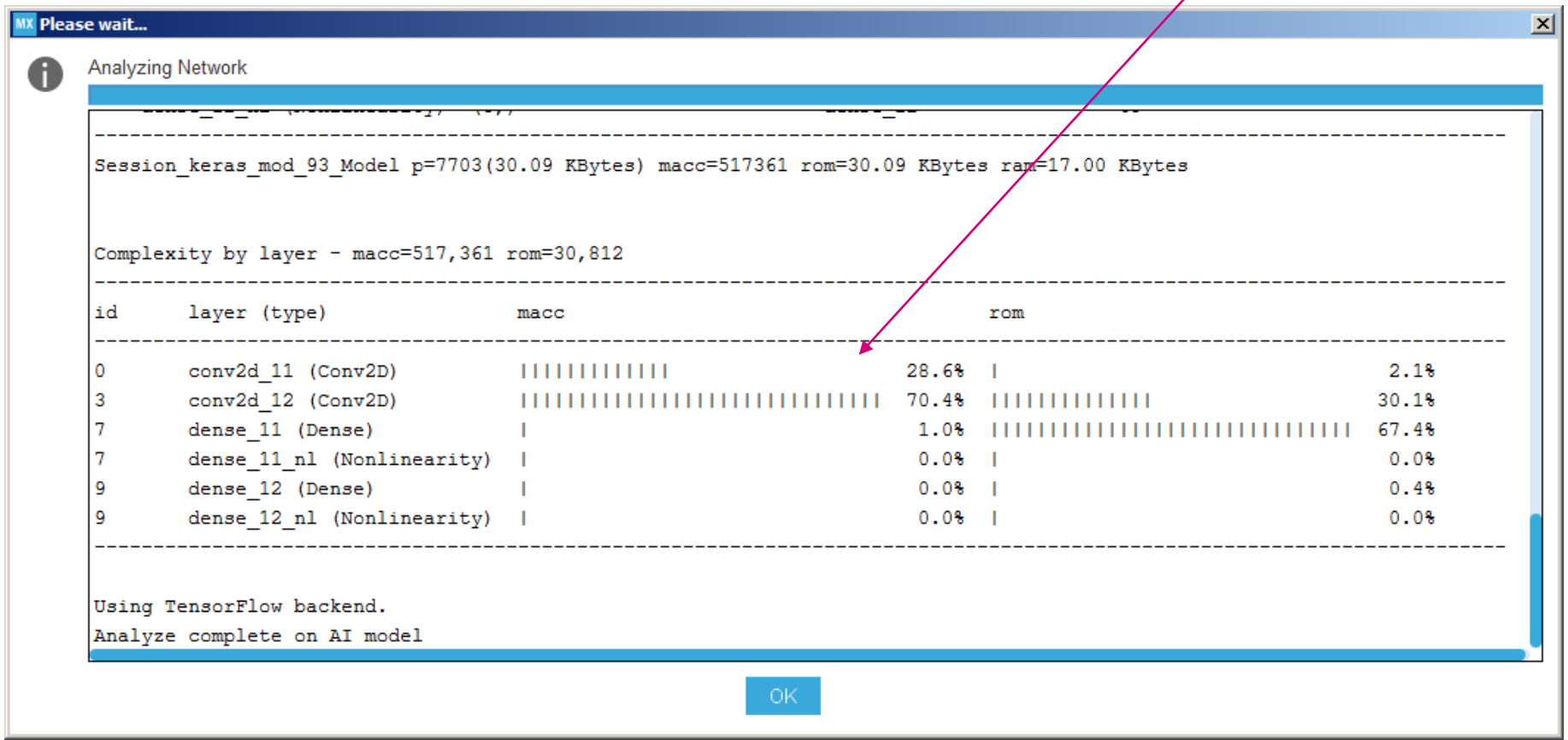
Validation outputs: None

Complexity: 517361 MACC
Flash occupation: 30.81 KBytes
RAM: 17.41 KBytes
Achieved compression: -
Analysis status: done

Evaluation status	Acc	RMSE	MAE
x86 C-model	-	-	-
stm32 C-model	-	-	-
original model	-	-	-
X-cross	-	-	-

Show graph

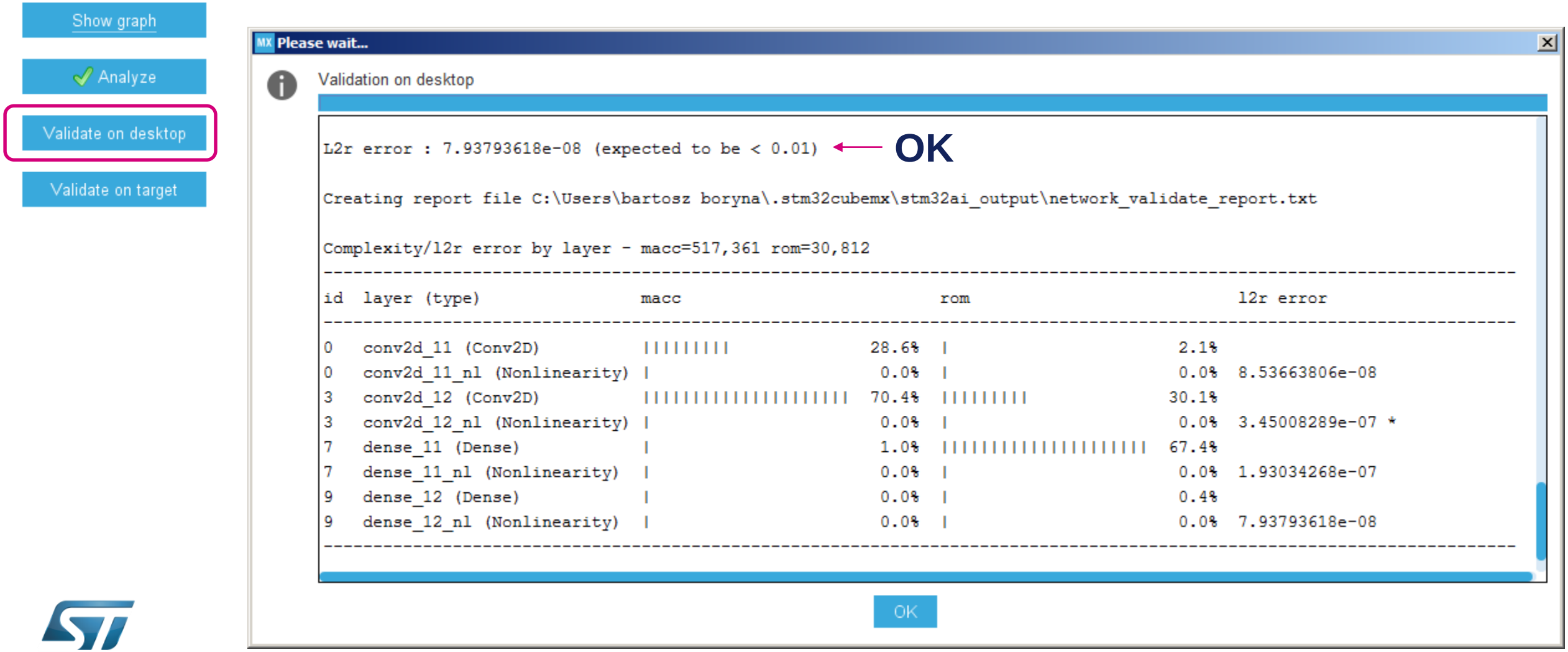
Analyze



life.augmented

Lab 3: X-CUBE-AI

Validate network on desktop, see result within **Output tab** and/or pop-up window

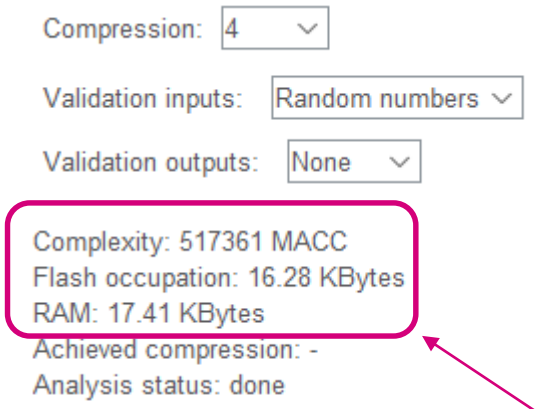


life.augmented

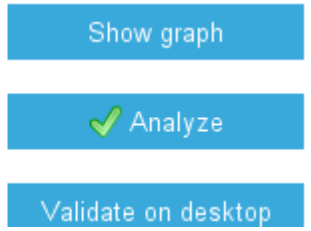
Lab 3: X-CUBE-AI

Try to save MCU resources using NN compression

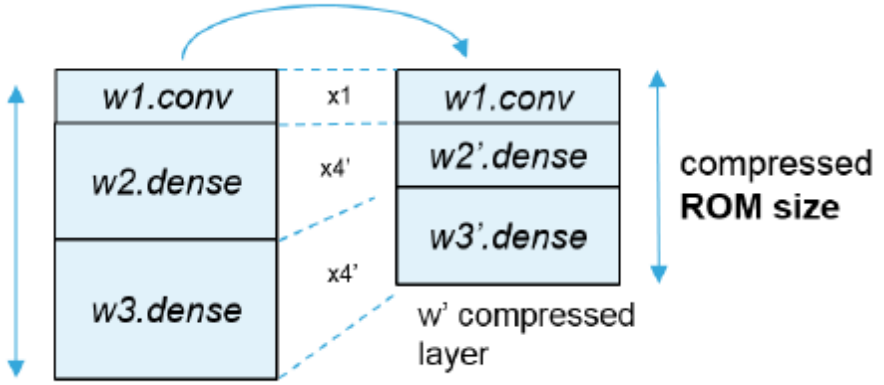
Set **Compression: 4**



Press **Analyze** button



Lower FLASH usage by ~50%

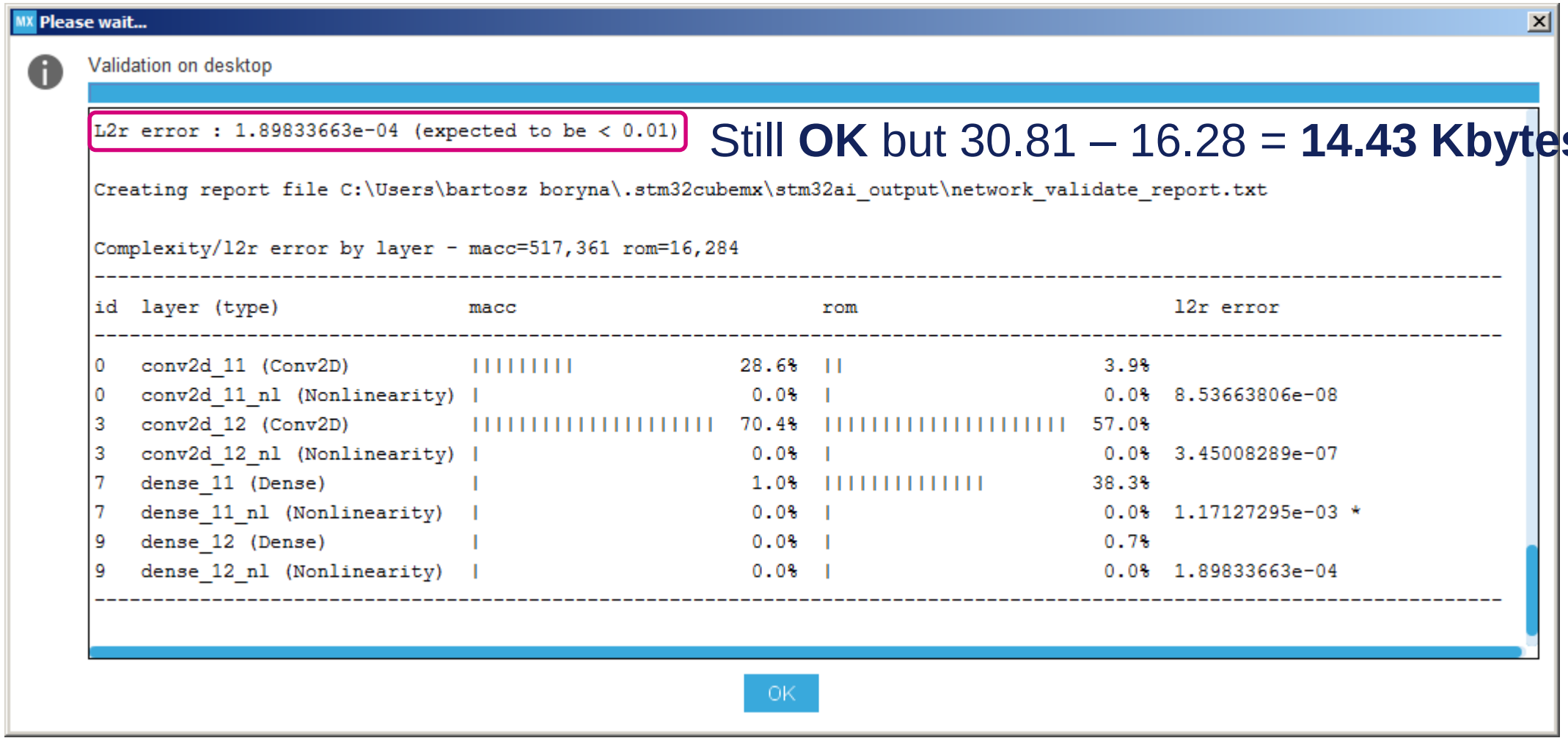


- Only applicable for dense (or fully-connected) layer type
- Weight-sharing-based algorithm is applied (K-means clustering)

Lab 3: X-CUBE-AI

Validate on desktop

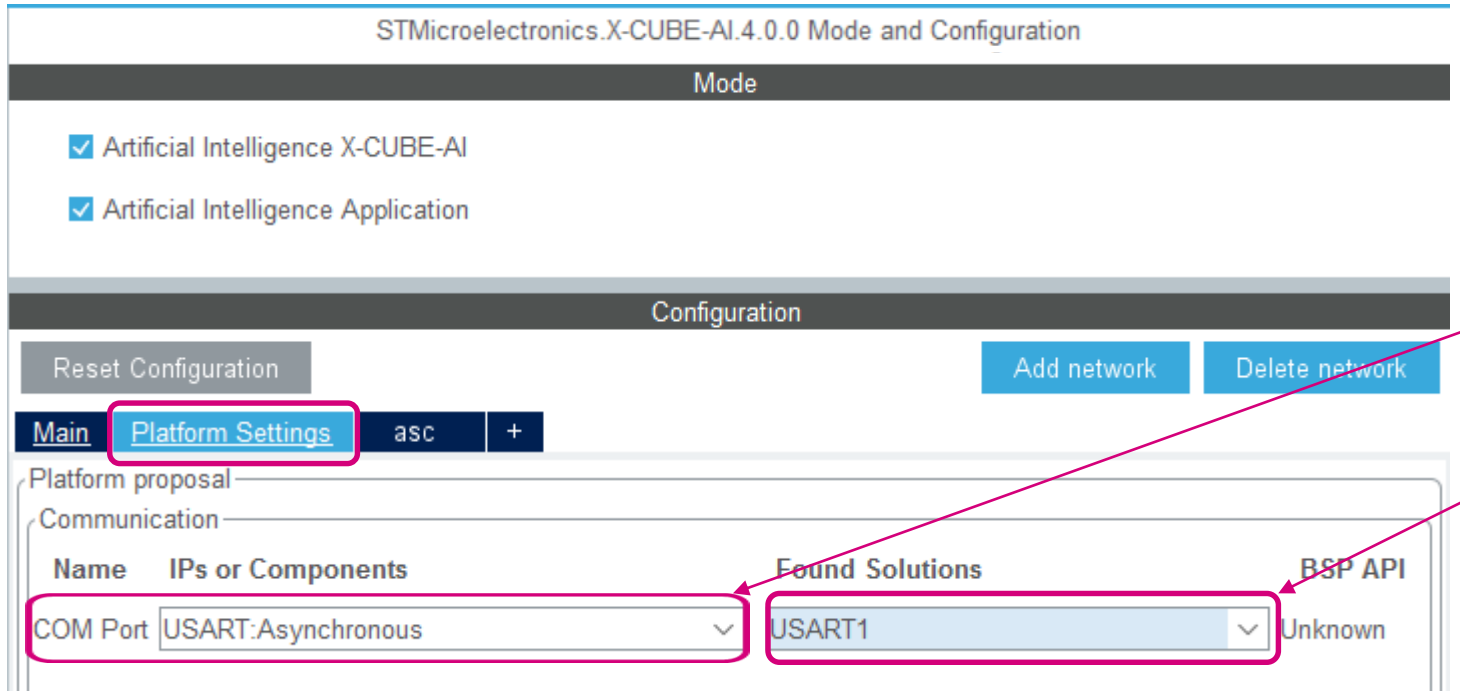
Validate compressed network on desktop, see result within **Output** tab



Still **OK** but 30.81 – 16.28 = **14.43 Kbytes saved**

Lab 3: X-CUBE-AI

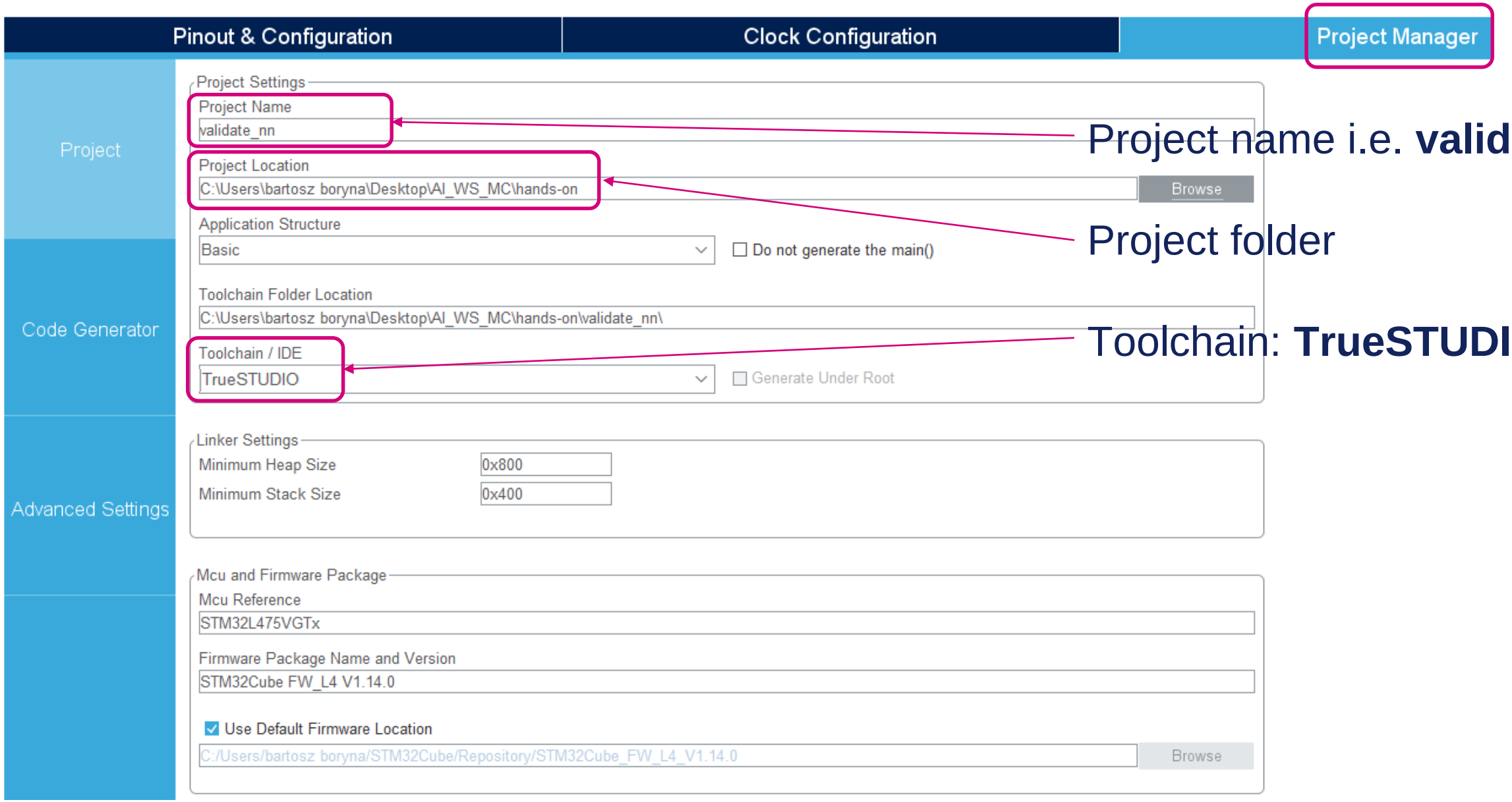
Configure application **Platform Settings**



Select **USART: Asynchronous**

Select **USART1** (connected to VCP of ST-Link)3

Project configuration



Project name i.e. **validate_nn**

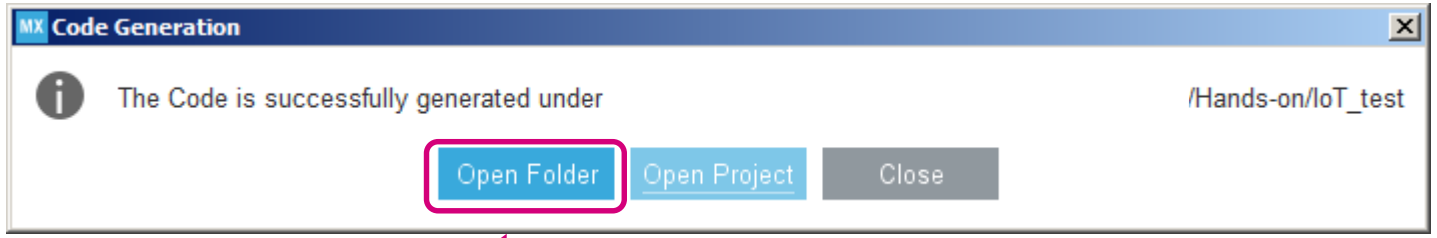
Project folder

Toolchain: **TrueSTUDIO**

Lab 3: X-CUBE-AI

Generate source code and open IDE

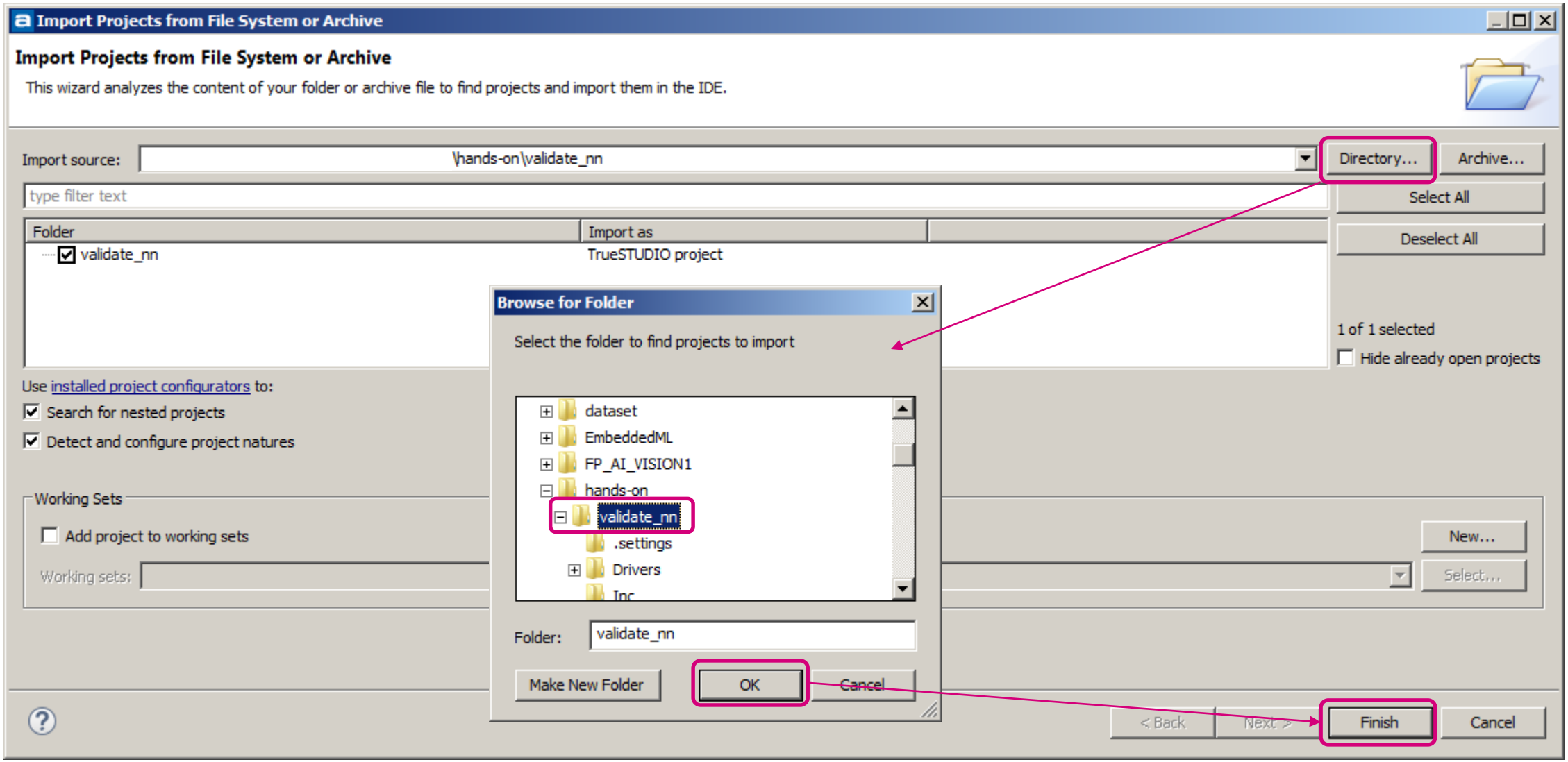
Press **GENERATE CODE** button



Open project

File → Open Projects from File System…

Lab 3: TrueSTUDIO



life.augmented

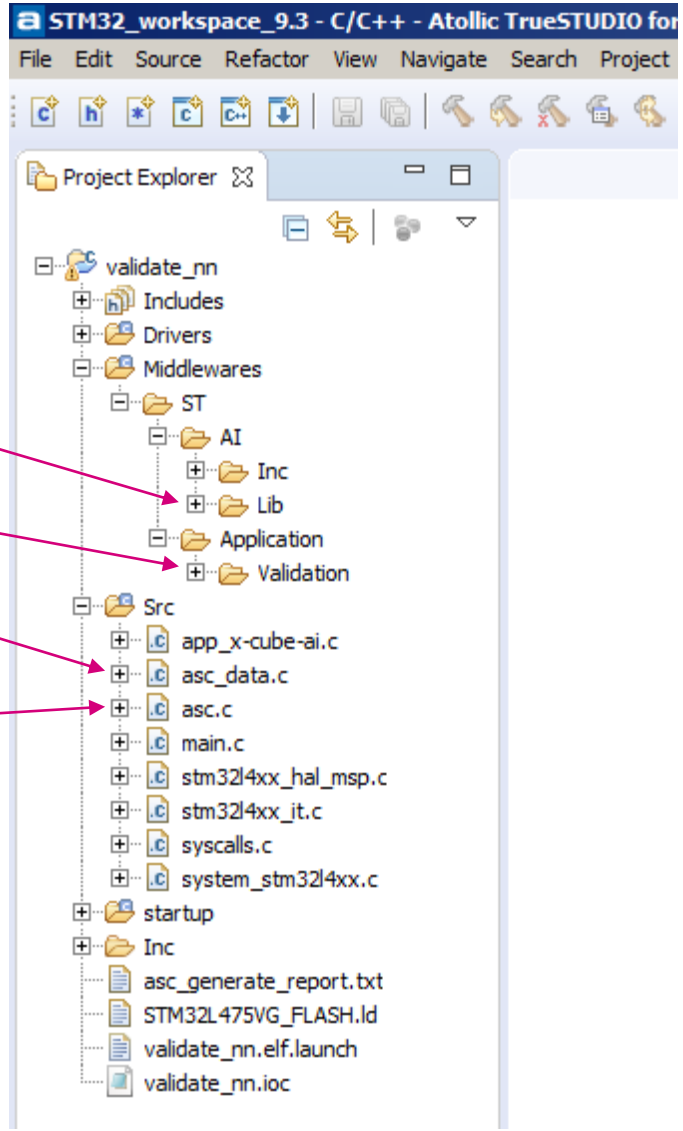
Lab 3: TrueSTUDIO

NN lib

NN app

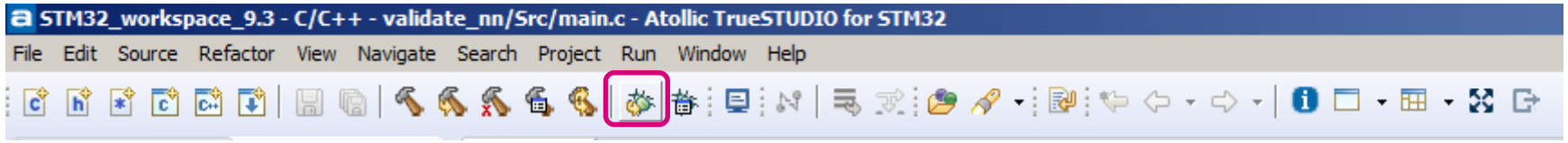
NN weights & biases

NN API

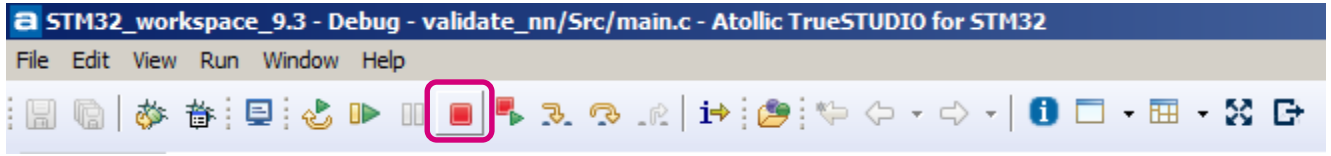


Lab 3: TrueSTUDIO

Build project and run Debugger



Stop Debugger just after flashing th binary



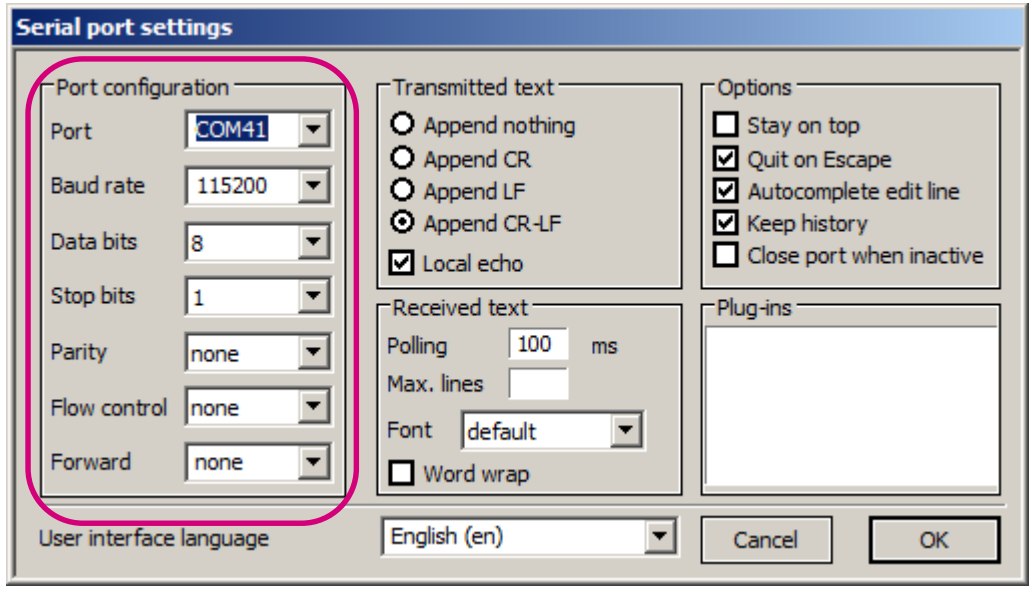

```
125
126   /* Infinite loop */
127   /* USER CODE BEGIN WHILE */
128   while (1)
129   {
130     /* USER CODE END WHILE */
131
132   MX_X_CUBE_AI_Process();
133     /* USER CODE BEGIN 3 */
134   }
135   /* USER CODE END 3 */
136 }
137
```

life.augmented

Lab 3: Validate on target

RESET IoT board (press black button)

Start terminal application, select COM port and set parameters: 115200,8,N,1



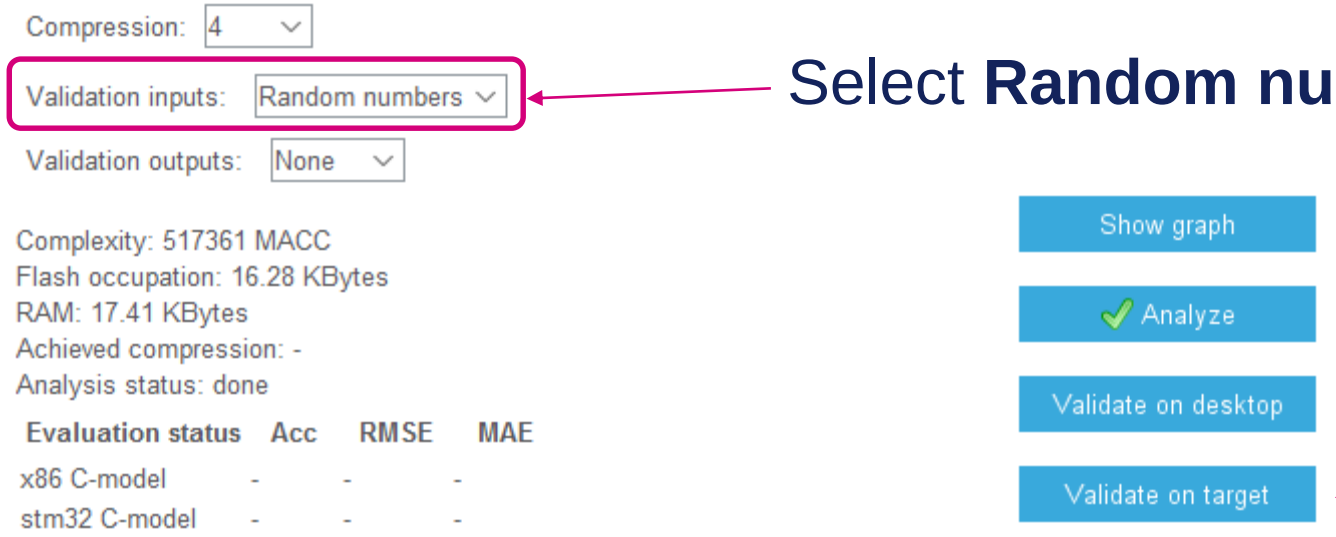
Close terminal / connection

```
#
# AI Validation 1.0
#
Compiled with IAR 9 (build 14835)
STM32 Runtime configuration...
 Device      : DevID:0x00000415 (STM32L4x6xx) RevID:0x00001007
 Core Arch.  : M4 - FPU PRESENT and used
 HAL version : 0x01090000
 system clock : 80 MHz
 FLASH conf. : ACR=0x00000604 - Prefetch=False $I/$D=(True,True) latency=4

AI platform (API 1.0.0 - RUNTIME 3.3.0)

Found network "network"
Creating the network "network"..
Network configuration...
 Model name       : network
 Model signature  : 0c4409ac77edf66cabac2e5ec04b2e65
 Model datetime   : Wed Apr 10 16:09:21 2019
 Compile datetime : Apr 10 2019 16:27:23
 Runtime revision : (3.3.0)
 Tool revision    : (rev-) (3.3.0)
Network info...
 nodes          : 6
 complexity     : 517361 MACC
 activation     : 18116 bytes
 weights        : 30812 bytes
 inputs/outputs : 1/1
 IN tensor format  : HWC layout:30,32,1 (s:960 f:AI_BUFFER_FORMAT_FLOAT)
 OUT tensor format : HWC layout:1,1,3 (s:3 f:AI_BUFFER_FORMAT_FLOAT)
Initializing the network

 ____________________________
| READY to receive a CMD from the HOST... |
 ____________________________

# Note: At this point, default ASCII-base terminal should be closed
# and a stm32com-base interface should be used
# (i.e. Python stm32com module). Protocol version = 1.0
```

life.augmented

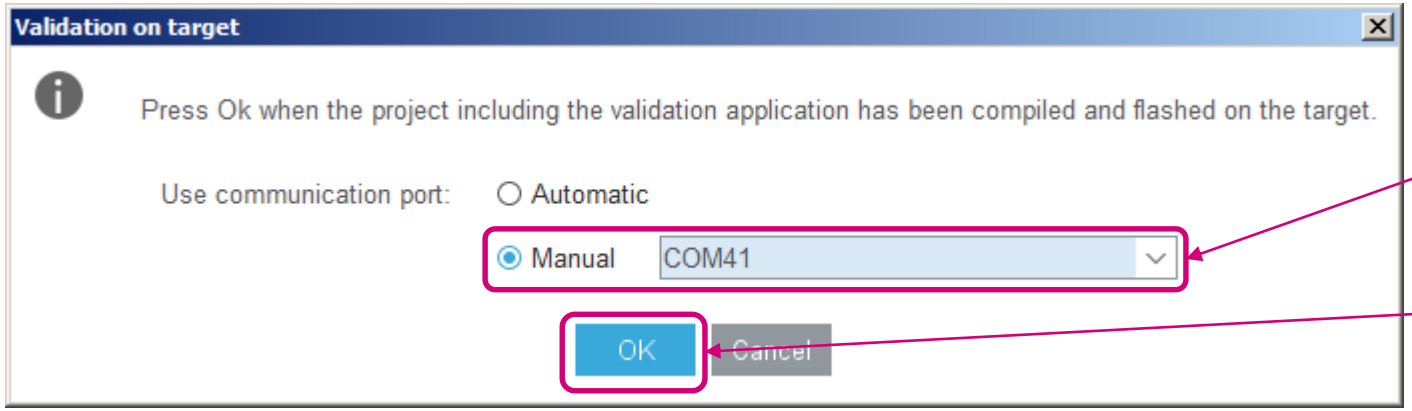
Lab 3: Validate on target

Come back to STM32CubeMX **Pinout & Configuration** tab



Select **Random numbers**

Press **Validate on target**

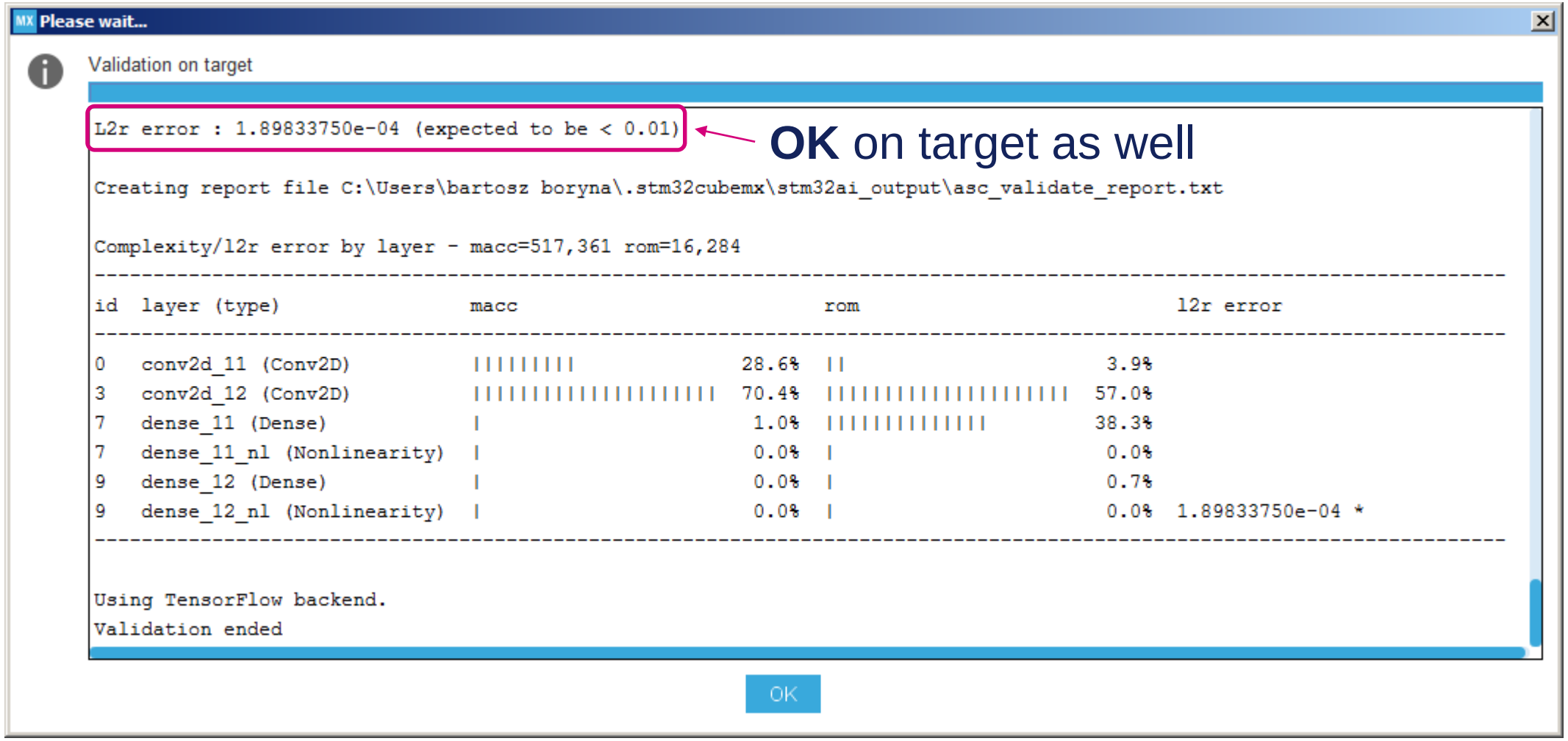


Select COM port manually

Press **OK**

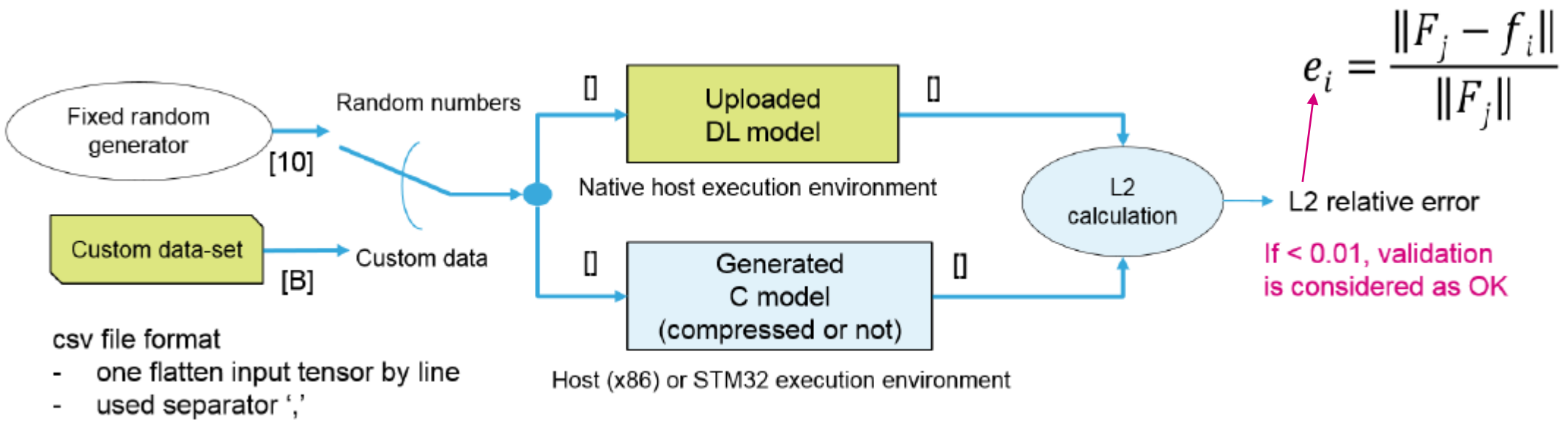
Lab 3: Validate on target

Analyze results in the log



NN validation

X-CUBE-AI **L2 relative error*** validation engine



life.augmented

* See UM2526 for details

Lab 3

Key messages

- X-CUBE-AI is an STM32CubeMX extension to convert pre-trained neural network models into optimized STM32 code.
- X-CUBE-AI also offers optimizations and validation tools dedicated to STM32
- Compression is a feature to reduce the network ROM footprint
- NN user code interface in `<network_name>.h`
- NN weights in `<network_name>_data.c`

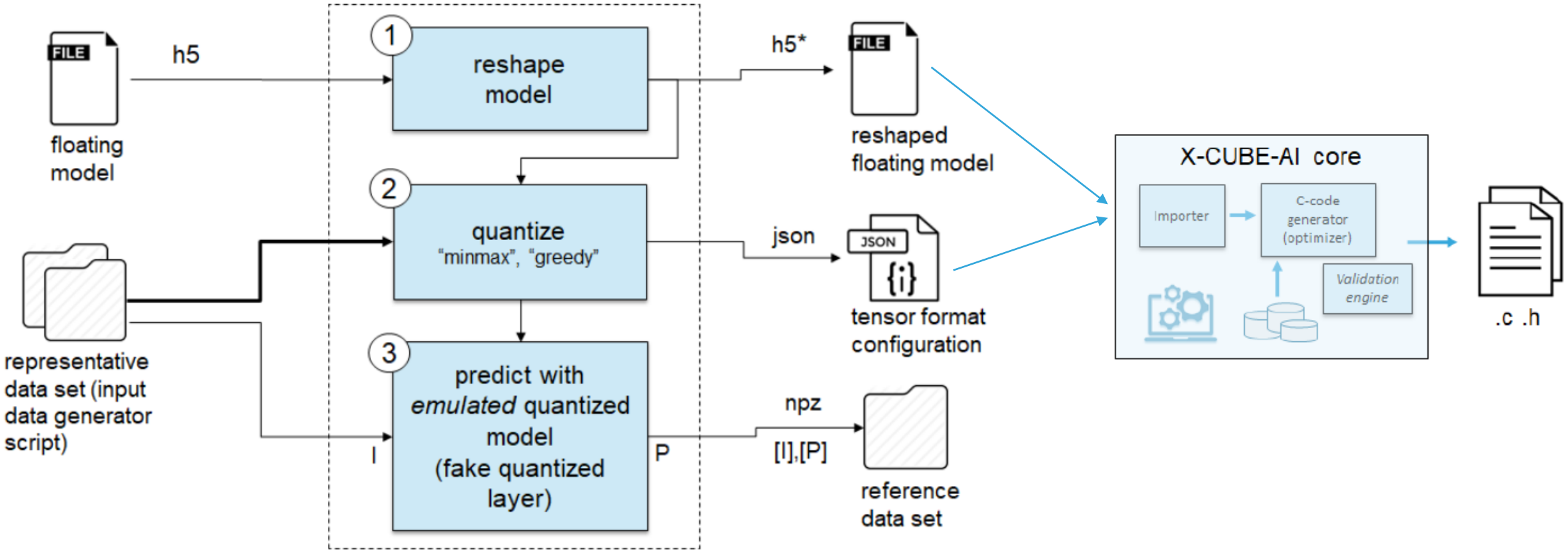
How to update modified NN model within your application

Key learning

- Update of FP-AI-SENSING1 software pack with previously generated `*.c / *.h` files consisting of **updated** NN model
- User test of updated NN behavior on target application / board

Lab 4

NN model quantization as efficient way to reduce MCU resources and decrease the inference time: the float format of weight and biases can be quantized to integer format



life.augmented

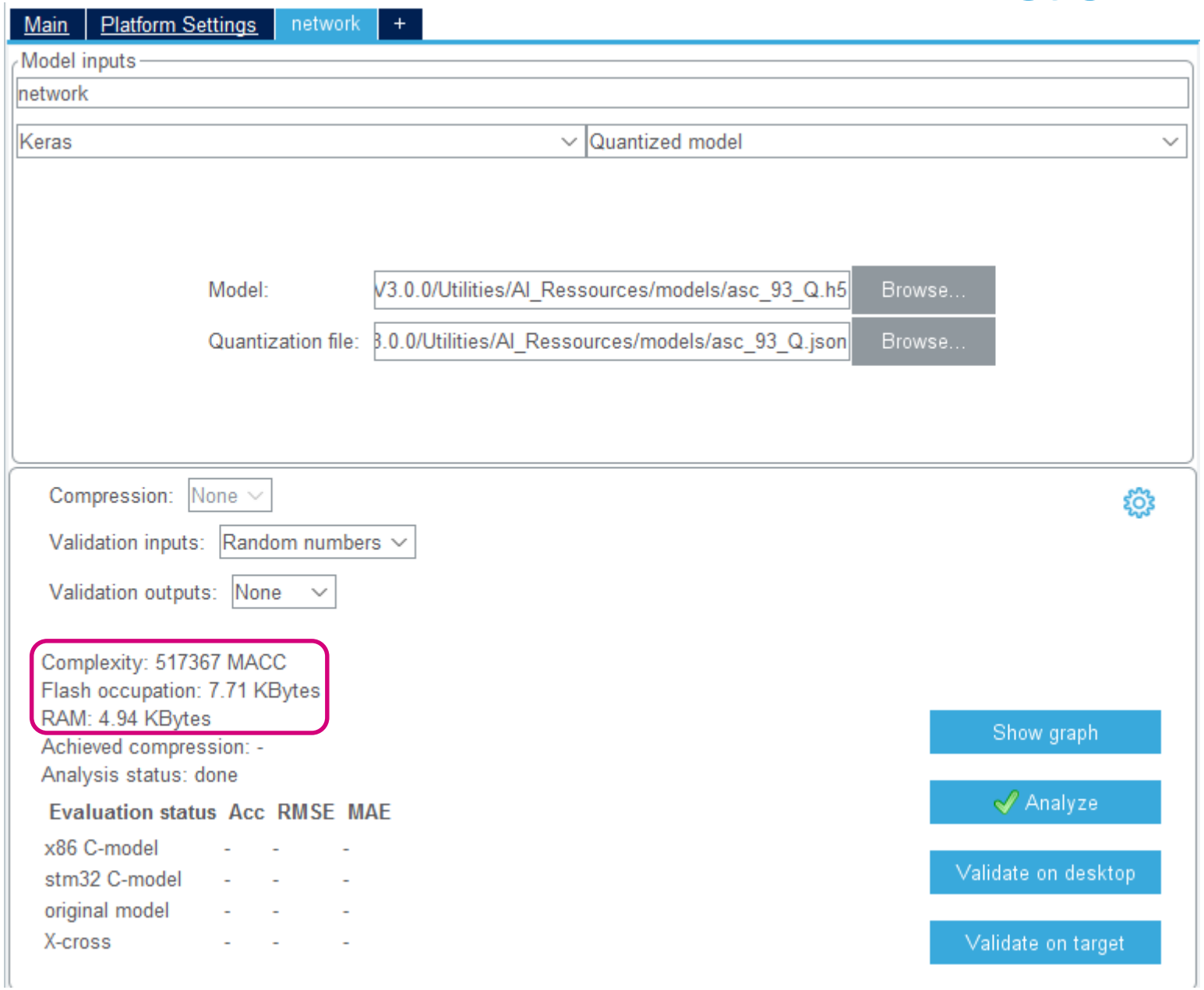
See details: …/STM32Cube/Repository/Packs/STMicroelectronics/X-CUBE-AI/4.0.0/Documentation/command_line_interface.html

Lab 4

Comparing to compressed ASC model

- Flash occupation reduced by **53%**
- RAM occupation reduced by **72%**

The cost of quantization is a little drop of NN model accuracy. The accuracy can be evaluated using reference data set.



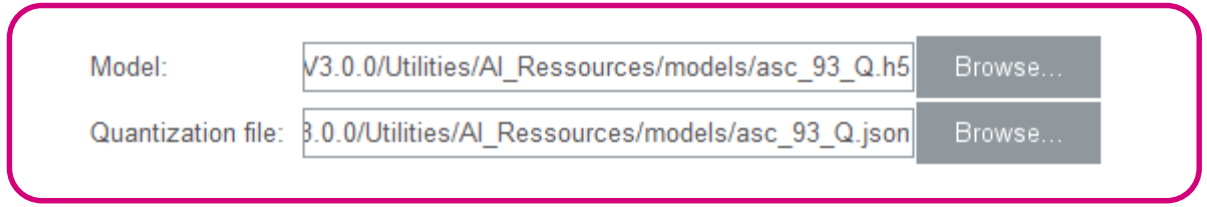
life.augmented

Configure an generate STM32CubeMX / X-CUBE-AI project following the **Lab3** steps

- As X-CUBE-AI v4.0.0 has been used to generated quantized C model for FP-AI-SENSING1, please select X-CUBE-AI v4.0.0

- Select “Template” application framework instead of “Validation”
- Select reshaped **asc_93_Q.h5** model and companion **asc_93_Q.json** tensor format file from below folder

C:\AI\FP-AI-SENSING1\Utilities\AI_Resources\models



Copy / Paste updated model

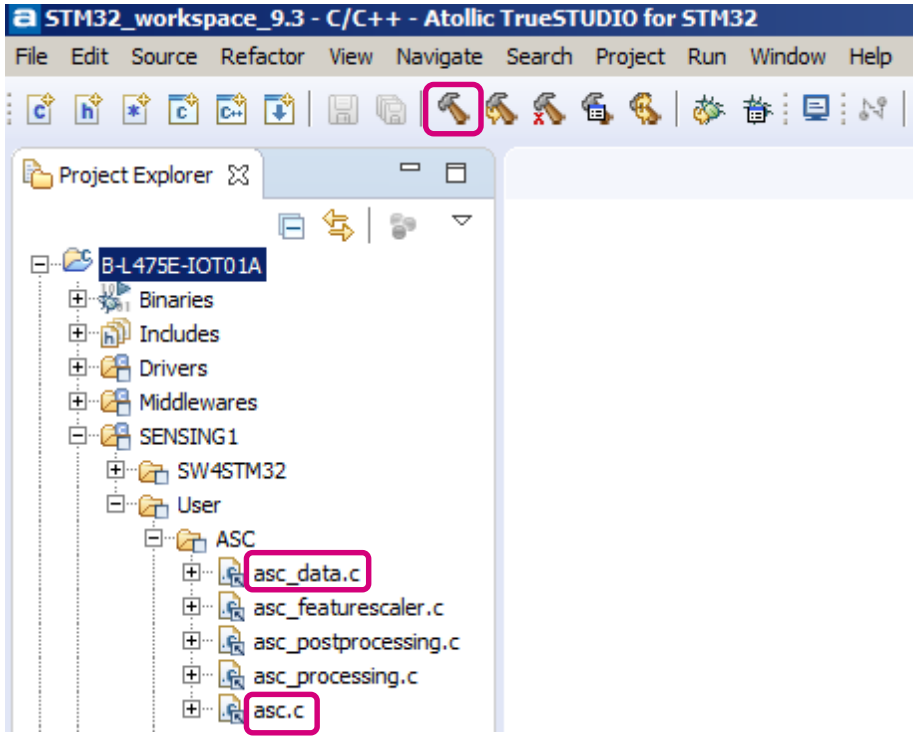
…update_nn\Src**asc_data.c**
Copy to C:\AI\FP-AI-SENSING1
\Projects\B-L475E-IOT01A\Applications\SENSING1\Src

…update_nn\Src**asc.c**
Copy to C:\AI\FP-AI-SENSING1
\Projects\B-L475E-IOT01A\Applications\SENSING1\Src

…update_nn\Inc**asc_data.h**
Copy to C:\AI\FP-AI-SENSING1
\Projects\B-L475E-IOT01A\Applications\SENSING1\Inc

…update_nn\Inc**asc.h**
Copy to C:\AI\FP-AI-SENSING1
\Projects\B-L475E-IOT01A\Applications\SENSING1\Inc

Lab 4: TrueSTUDIO



Open & Build **FP-AI-SENSING** project

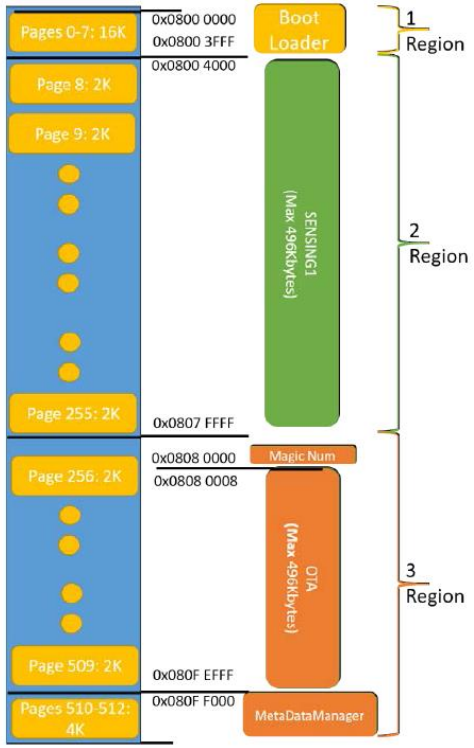
File → Open Projects from File System…

C:\AI\FP-AISENSING1\Projects
\B-L475E-IOT01A\Applications\SENSING1\SW4STM32
\B-L475E-IOT01A

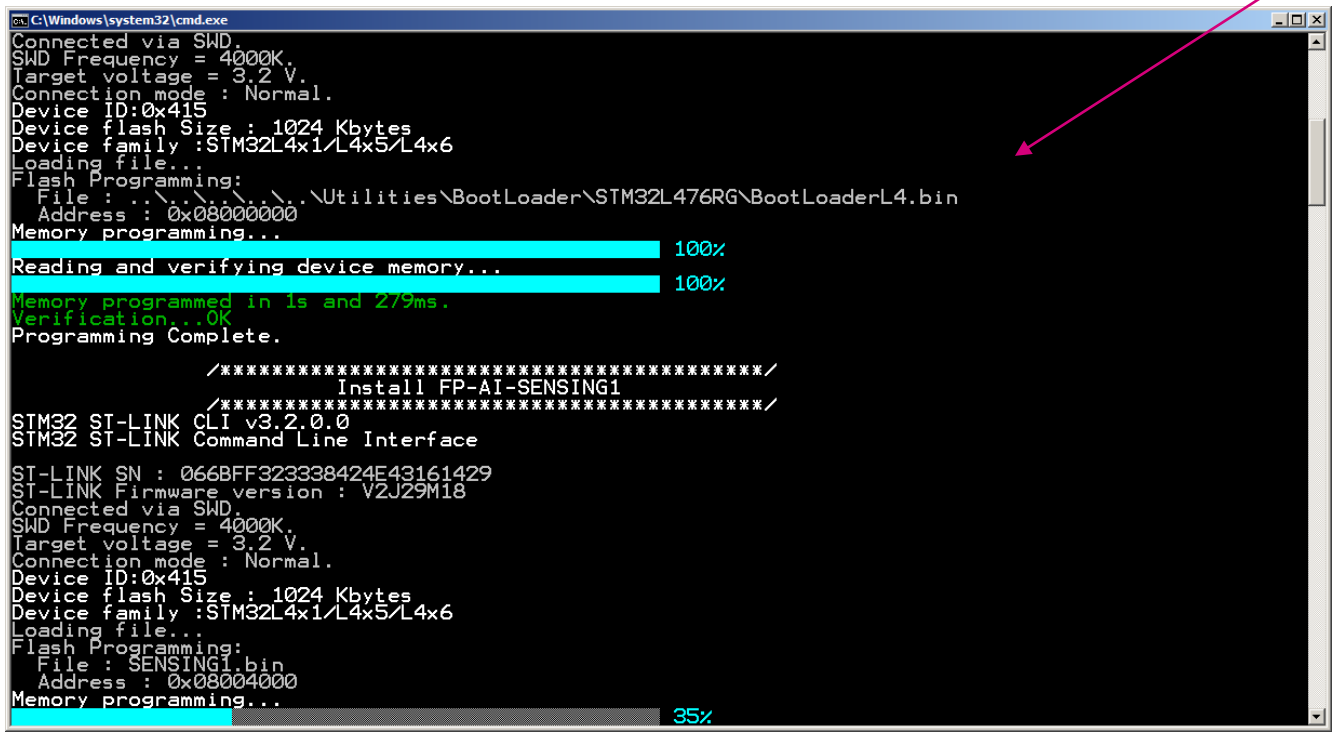
Download application

Due to dedicated FLASH memory allocation both for custom bootloader and application, there is dedicated script to erase chip, download bootloader and application, see UM2524 for details.

Figure 3. **FP-AI-SENSING1 Flash structure**



C:\AI\FP-AI-SENSING1\Projects\B-L475E-IOT01A\Applications\SENSING1\SW4STM32**CleanSENSING1.bat**



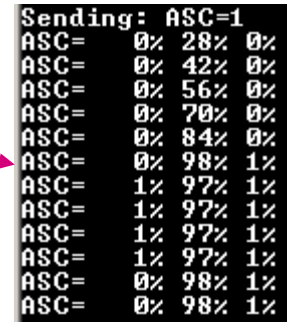
Follow dialogs on Windows command line

life.augmented

Lab 4: Terminal

Test updated FP-AI-SENSING1 app using STBLESensor app , see lab 1

Console interface is also possible: start terminal (115200,8,N,1), and then press **reset button** on IoT board, enter <help> command and follow printed out help to start ASC application: enter <asc start> command.



Lab 4

Key messages

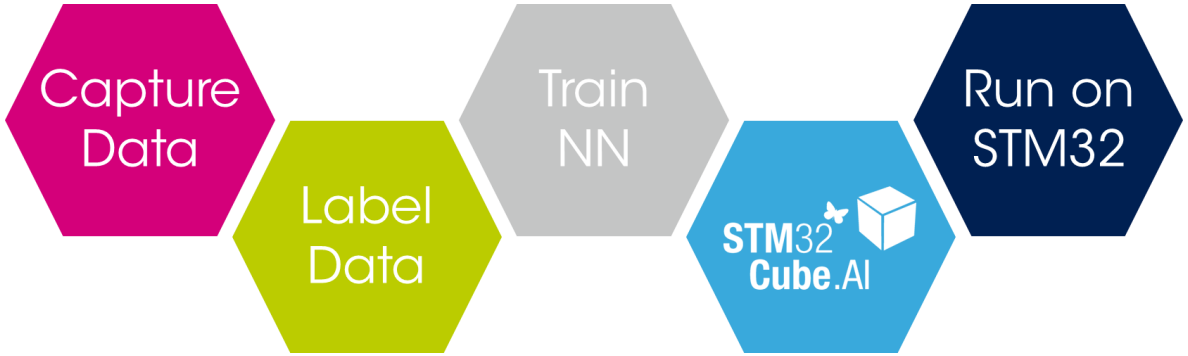
- The NN model implantation generated by X-CUBE-AI is defined in :
  - `<network_name>.h`
  - `<network_name>.c`
  - `<network_name>_data.h`
  - `<network_name>_data.c`
- These files can be copy-pasted into FP-AI-SENSING1

Conclusion

- WE have demonstrated easiness of NN application development using STM32CubeAI
  - Configuration and generation of pre-trained NN based software project
  - Update of NN model within custom project using STM32CubeMX tool
  - Validation of real NN model on target board
- YOU will be in a position to understand and implement it

For more Information

www.st.com/STM32CubeAI

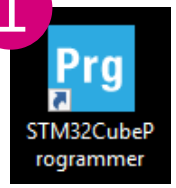


Upgrade of ST-Link FW

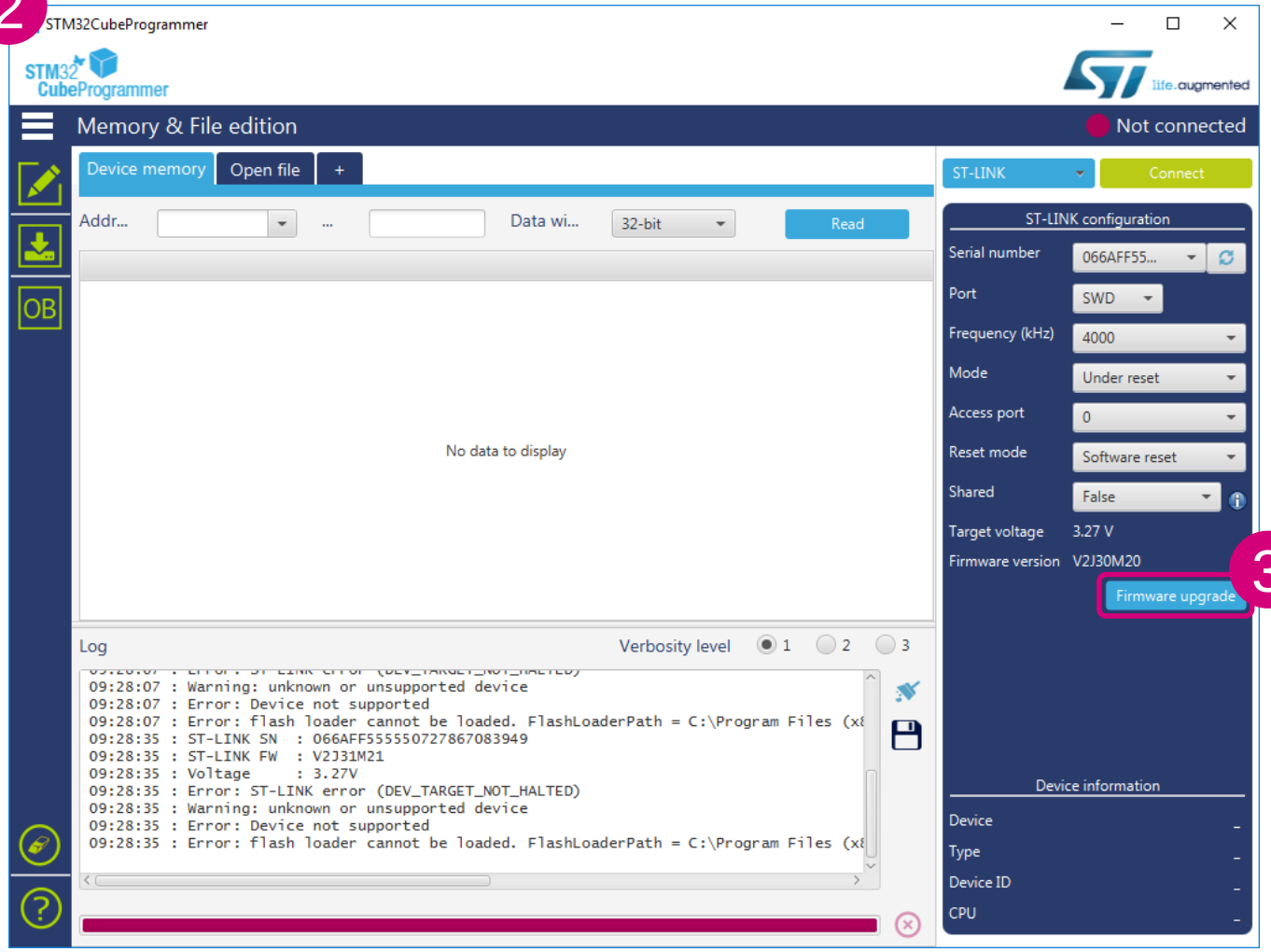
Upgrade the on-board ST-Link FW 1/2

- Connect the nucleo board to your PC

1



2



3

Firmware upgrade

life.augmented

STM32 WB

Upgrade the on-board ST-Link FW 2/2

Click Refresh device list

Click Open in update mode

Check firmware version

Click Upgrade (if newer available)

Close the window and STM32CubeProgrammer afterwards.

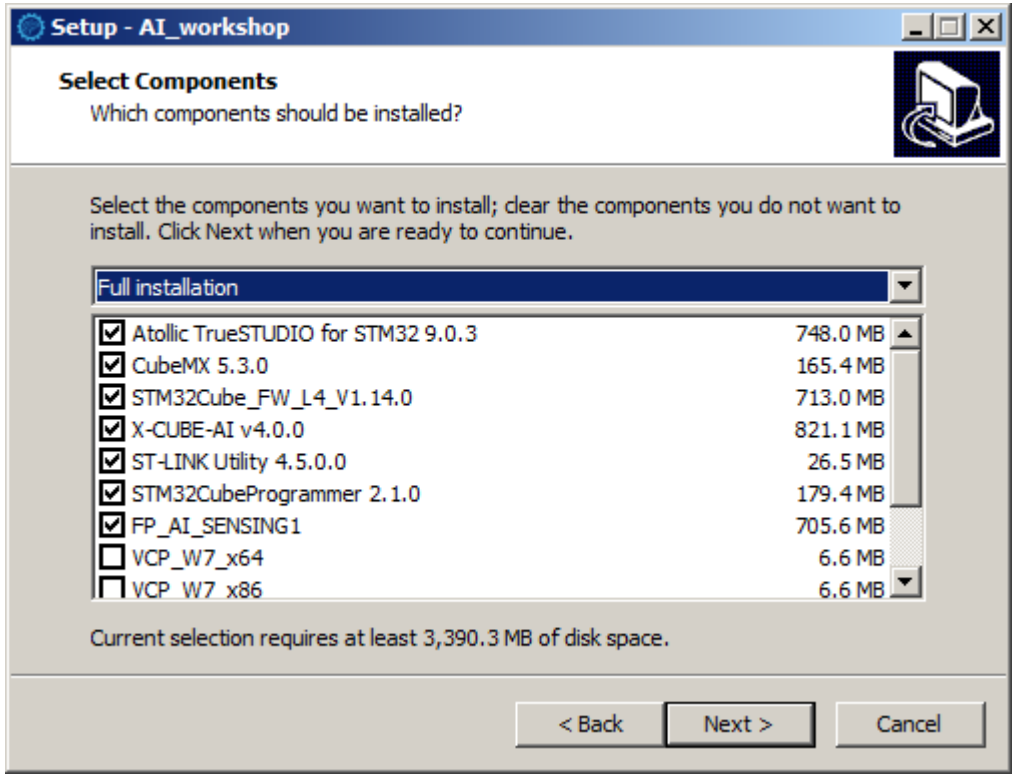
life.augmented

STM32 WB

Attendee preparation

Prerequisites

1. Download and start **AI_workshop_1.0.exe / -1a.bin / -1b.bin** tools installer

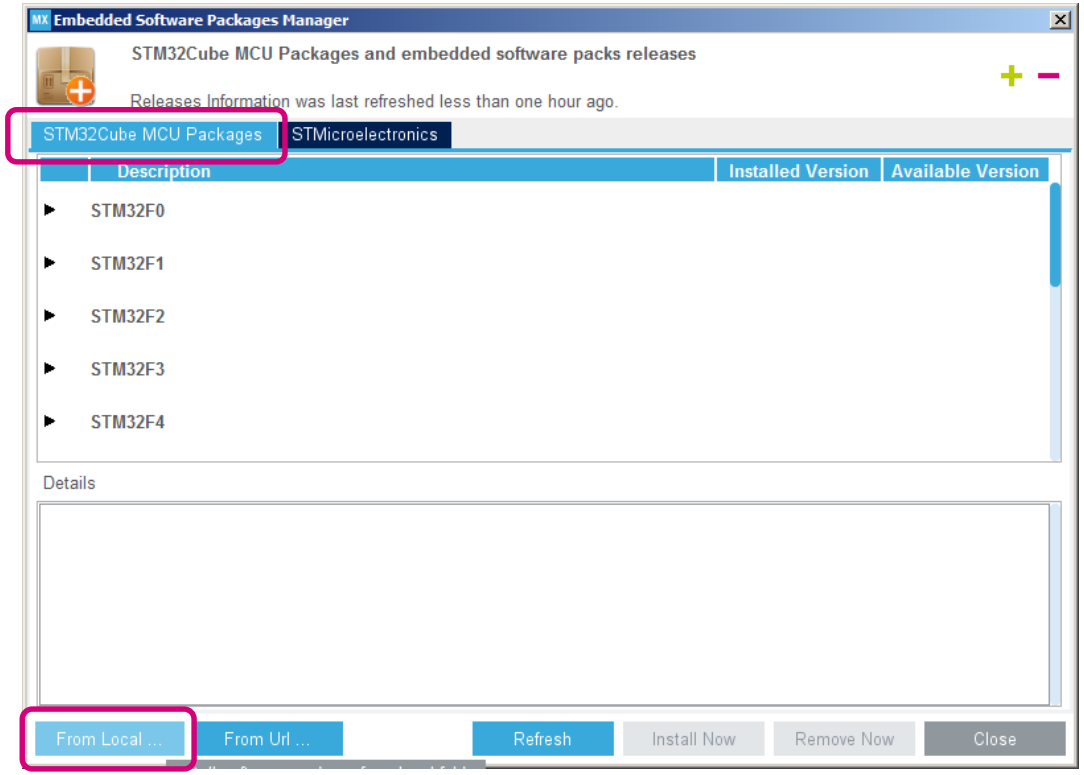


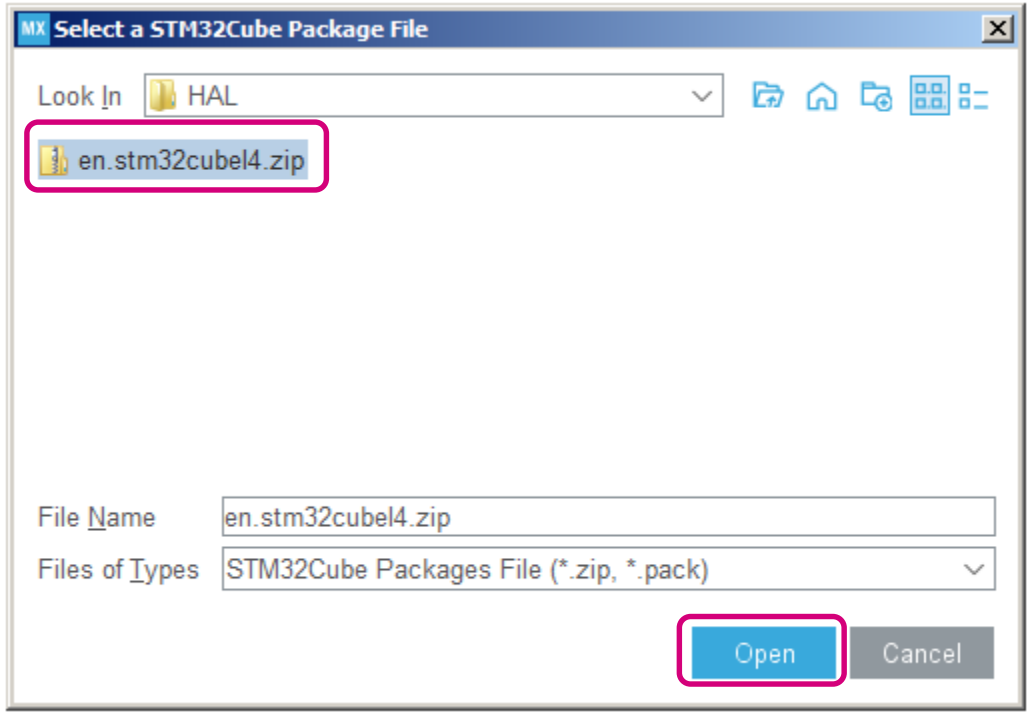
Installer components:

- Attolic TrueSTUDIO v9.3.0
- STM32CubeMX v5.3.0
- STM32Cube_FW_L4_V1.14.0
- X-CUBE-AI v4.0.0
- ST-LINK Utility v4.5.0
- STM32CubeProgrammer v2.1.0
- FP-AI-SENSING1 v3.0.0
- Virtual Com Port drivers (option)

2. Start STM32CubeMX and install*
   STM32Cube_FW_L4_V1.14.0 library

Help → Manage embedded software packages
→ STM32Cube MCU Packages tab **→ From Local…**
→ Open C:\AI\HAL**en.stm32cubel4.zip**





life.augmented

*See also section "**Installing STM32 MCU packages**" of UM1718, link to document is below
https://www.st.com/content/ccc/resource/technical/document/user_manual/10/c5/1a/43/3a/70/43/7d/DM00104712.pdf/files/DM00104712.pdf/jcr:content/translations/en.DM00104712.pdf

Prerequisites

3. Extract **C:\Package\en.x-cube-ai-v4-0-0.zip** file

4. Install X-CUBE-AI embedded software pack*
Help → Manage embedded software packages
→ **STMicroelectronics** tab → **From Local…**
→ **Open** C:\AI\Package**STMicroelectronics.X-CUBE-AI.4.0.0.pack**

5. Download and install terminal application i.e. Tera Term if not already installed

6. Download and install on your mobile phone or tablet **STBLESensor** application, follow the expected OS of your device (Android/IOS), link is below

https://www.st.com/content/st_com/en/products/embedded-software/wireless-connectivity-software/stblesensor.html

life.augmented

*See also section “Installing embedded software packs” of UM1718, link to document is below
https://www.st.com/content/ccc/resource/technical/document/user_manual/10/c5/1a/43/3a/70/43/7d/DM00104712.pdf/files/DM00104712.pdf/jcr:content/translations/en.DM00104712.pdf