life.augmented

Converting Neural Networks model into Optimized Code for MCU: STM32Cube.AI tool

Amit Kumar

5th Aug 2021

STM32 Cube.AI

life.augmented

STM32 comprehensive AI ecosystem

Applications

Applicative Examples
(Function Packs)

Pre and post Processing libraries

Frameworks

NANOEDGE AI STUDIO

scikit learn

MATLAB

PyTorch

TensorFlow Lite

ONNX

Keras

STM32 Cube.AI

AI Model convertor

Graph optimizer | Quantizer | Memory optimizer

Edge Hardware

STM32 series

Discovery kit

STM32 Nucleo board

Camera add-on

life.augmented

A tool to seamlessly integrate AI in your projects

Machine Learning

scikit learn

ONNX

Deep Learning

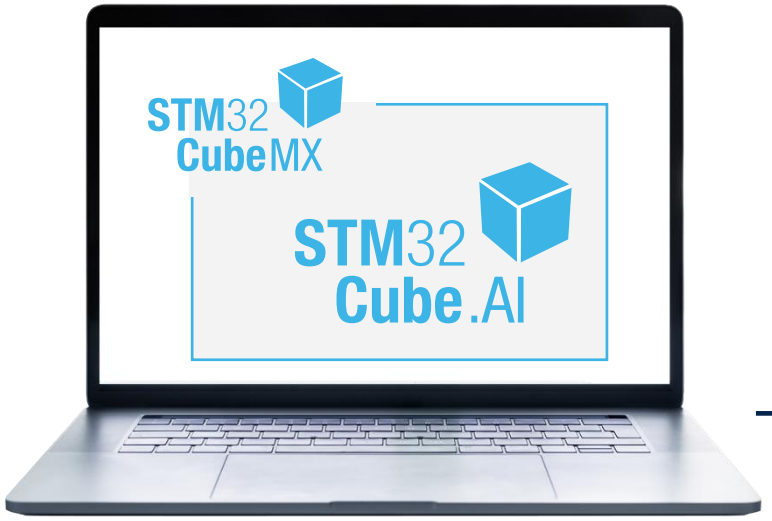
MATLAB

PyTorch

TensorFlow Lite

Keras

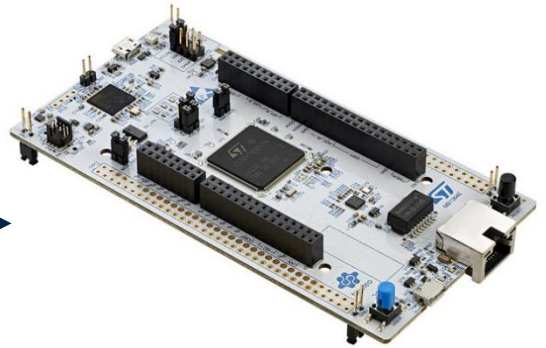
ONNX



Select MCU & upload your model

Optimize and validate

Generate project and deploy



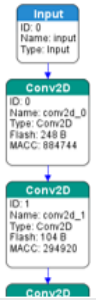
HIGH PERFORMANCE EDGE AI PRODUCT

life.augmented

The 3 pillars of STM32Cube.AI

Graph optimizer

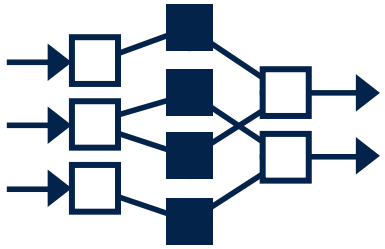
Automatically improve performance through graph simplifications & optimizations that benefit STM32 target HW architectures



- Auto graph rewrite
- Node/operator fusion
- Layout optimization
- Constant-folding…
- Operator-level info to fine-tune memory footprint and computation

Quantized model support

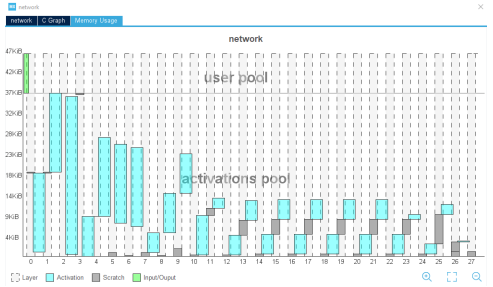
Import your quantized ANN to be compatible with STM32 embedded architectures while keeping their performance



- From FP32 to Int8
- Minimum loss of accuracy
- Code validation on target
  - Latency
  - Accuracy
  - Memory usage

Memory optimizer

Optimize memory allocation to get the best performance while respecting the constraints of your embedded design



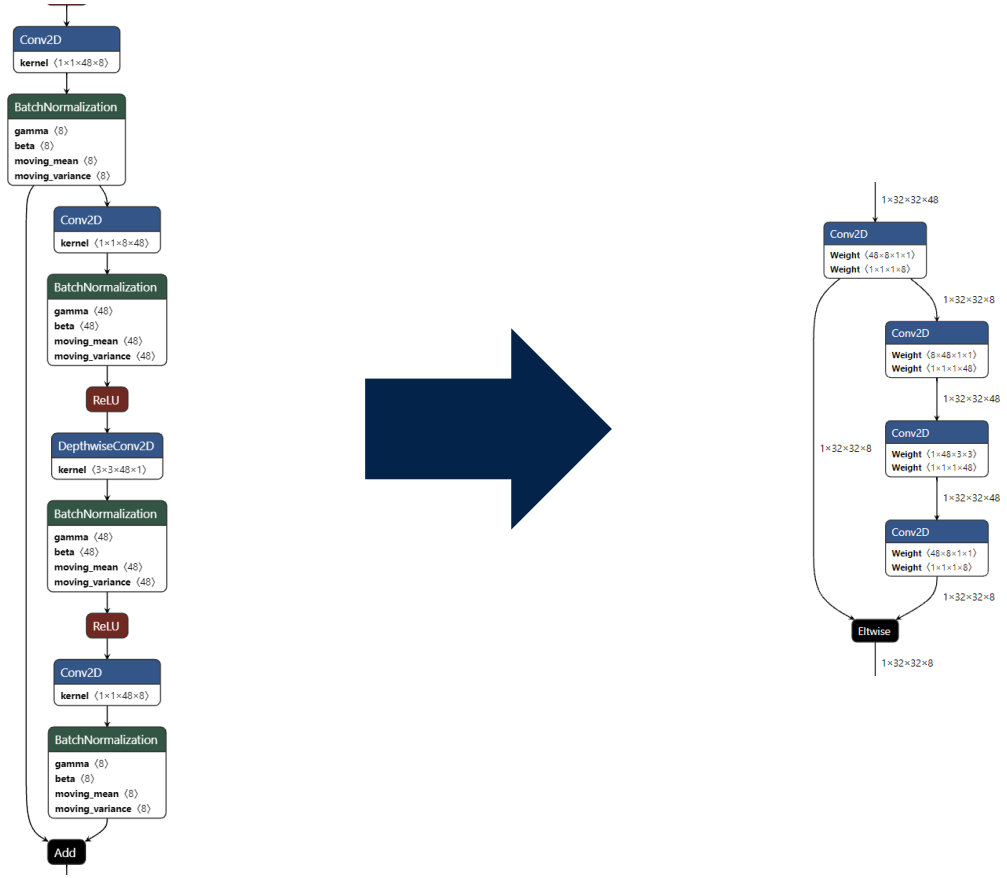
- Memory allocation
- Internal/external memory repartition
- Model-only update option

STM32Cube.AI is **free of charge**, available both in graphical interface and in command line.

life.augmented

Graph optimizer

Squeeze your graph to fit into an MCU!



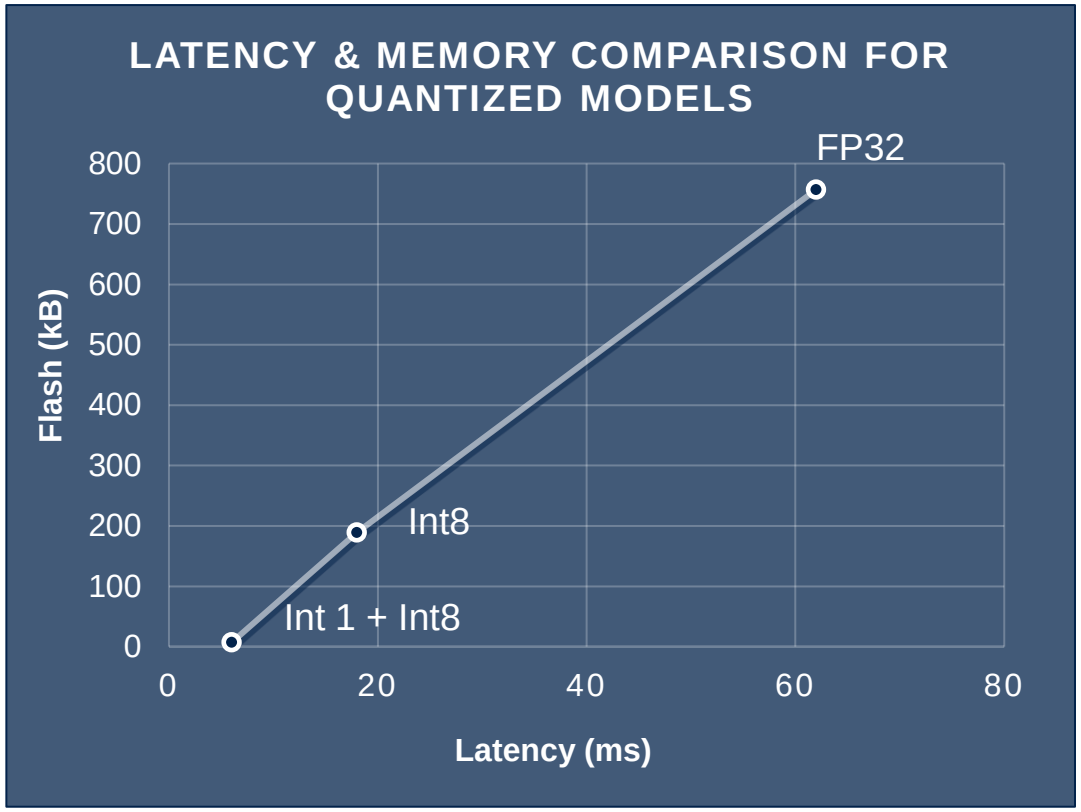
Fully automated process in the STM32Cube.AI workflow

- Your original graph is optimized at the very early stage for optimal integration into STM32 MCU/MPU
- Loss-less conversion

life.augmented

Quantized model support

Simply use quantized networks to reduce memory footprint and inference time



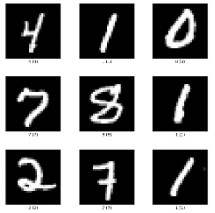
STM32Cube.AI support quantized Neural Network models with **all parameter formats**:

- 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

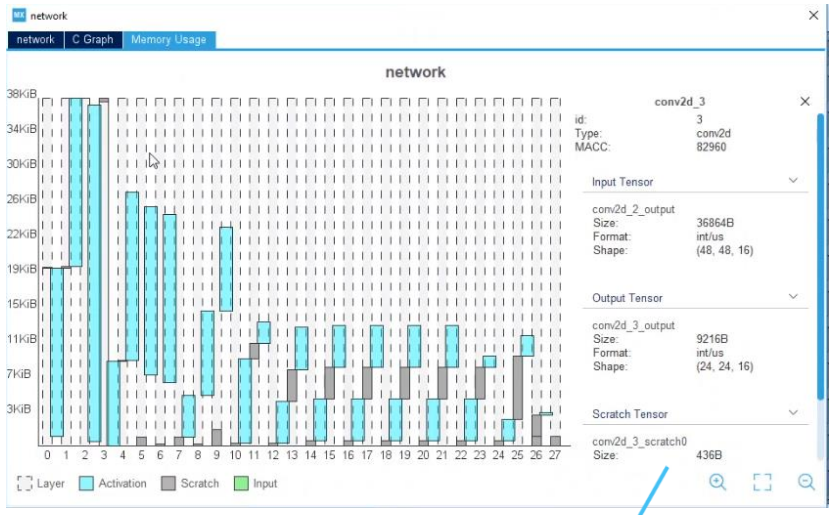


MNIST dataset

life.augmented

Memory optimizer

Optimize performance easily with the memory allocation tool

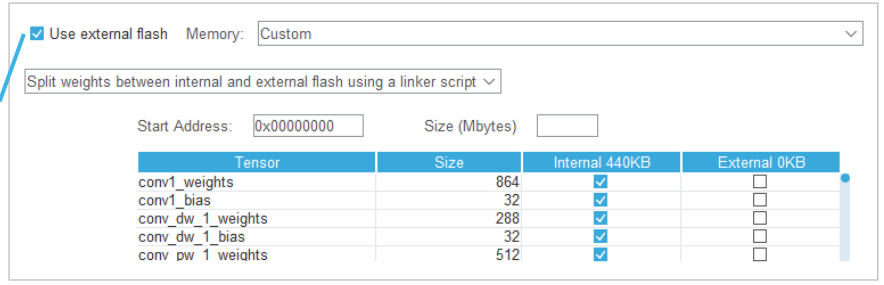


Model RAM consumption per layer

- Easily identify most critical layers

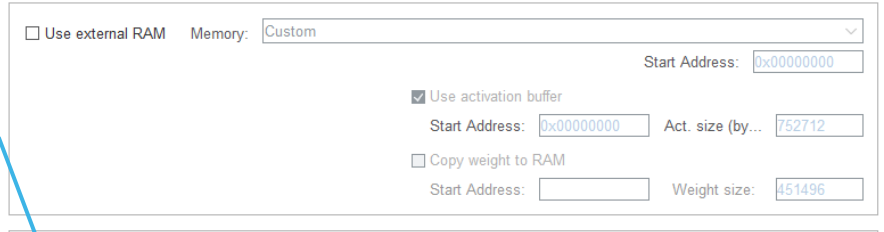
Model memory allocation

- Set your external memory
- Map in non-contiguous internal flash section
- Partition internal vs external flash memories



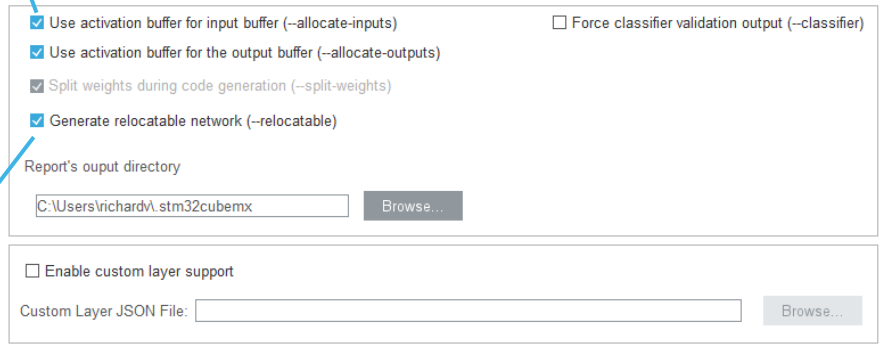
Re-use model input buffer to store activation data*

- Minimize RAM requirements



Relocatable network

- A separate binary is generated for the library and the network to enable standalone model upgrade

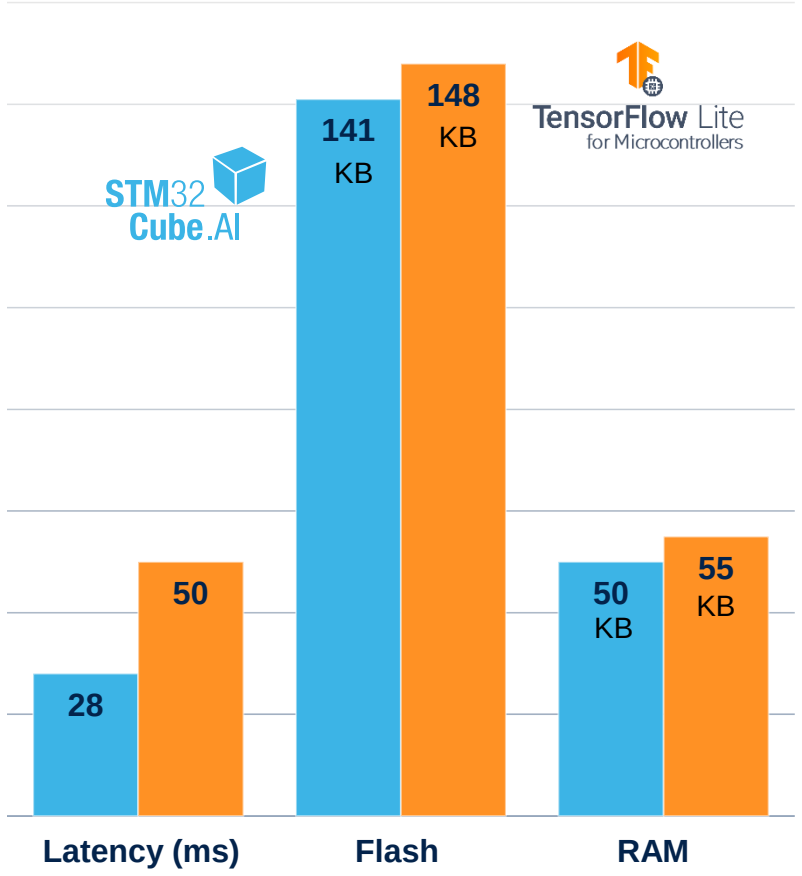


** Requires input and activation buffers in same memory*

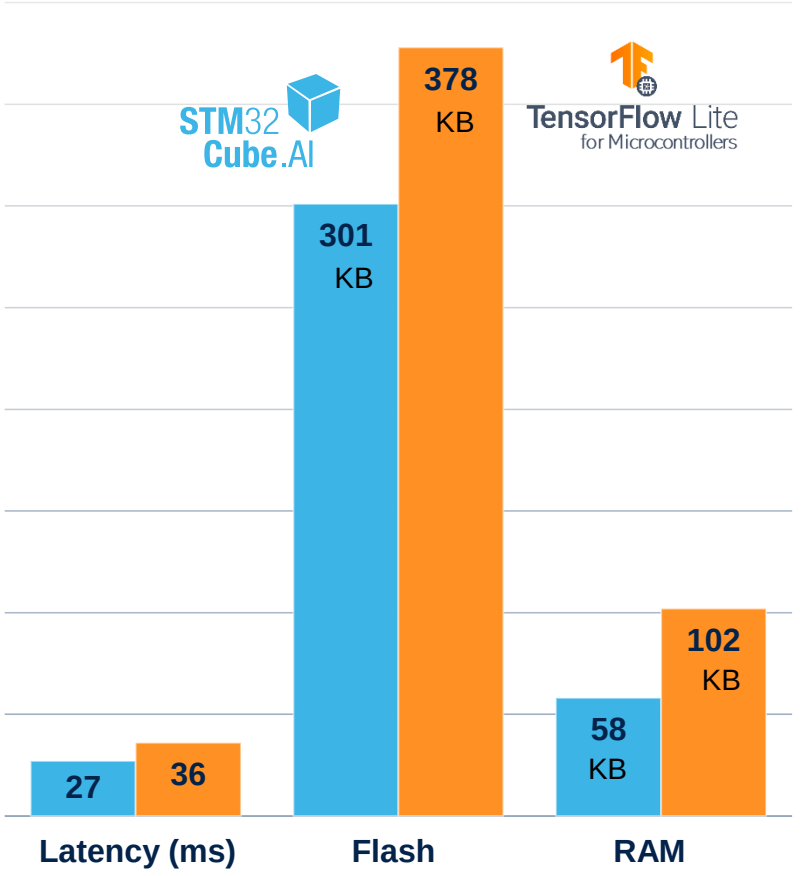
life.augmented

STM32Cube.AI
Get the best performance on STM32

Image Classif v0.7



Visual Wake Word v0.7



HW Target: STM32H723
Flash: 1Mbyte
RAM: 564 Kbytes
Freq: 550 MHz

SW Version:
X-Cube.AI v 7.2.0
TFLm v2.7.0

** the lower the better*

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

STM32F3
245 CoreMark
72 MHz Cortex-M4

STM32G4
569 CoreMark
170 MHz Cortex-M4

Mixed-signal MCUs

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

What's new in STM32Cube.AI v7.2.0?

STM32Cube.AI development tool supports deeply quantized neural networks

Further reduce AI code with deep quantization

- Introducing support for mixed precision quantization and binary neural network (BNN) for STM32
- Supporting pre-trained quantized models from:
  - qKeras
  - Larq

Improved performance tuning

- Addition of a new kernel performance enhancement for further optimization of memory footprint and power consumption

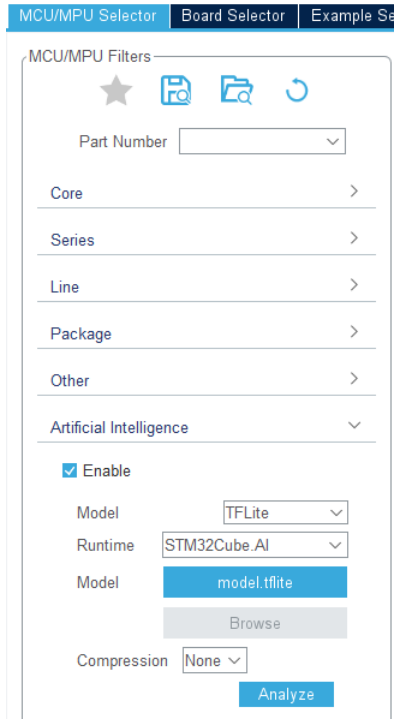
Up-to-date and improved code generation

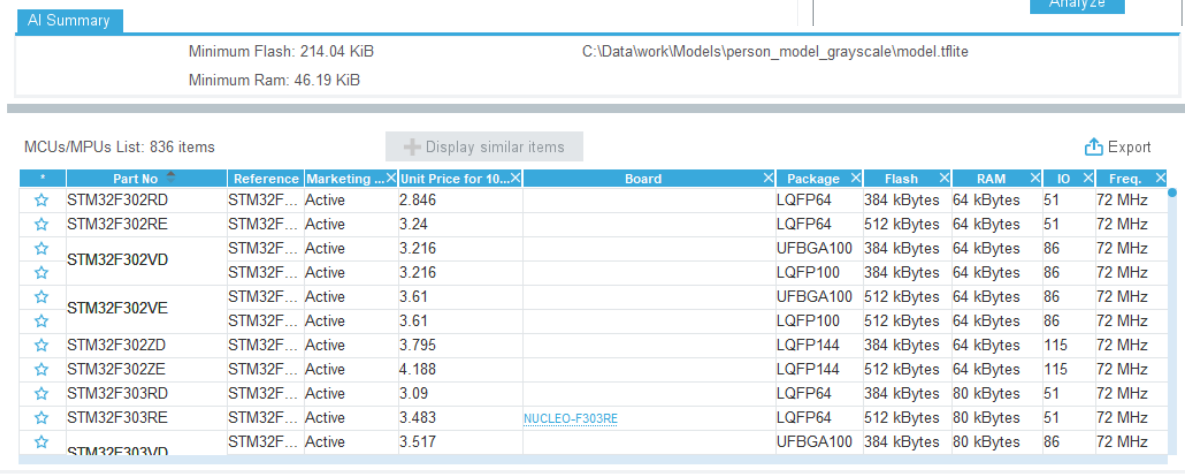
- Support TensorFlow v2.9 models
- Support of new ONNX operators (refer to documentation for exhaustive list)
- Extend support of scikit-learn ML algorithms

STM32Cube.AI user flow (1/3)

1. Select MCU
2. Optimize and validate
3. Generate project and integrate

- Load model and select an AI runtime
- Analyze minimal required footprint
- Select corresponding STM32 MCUs



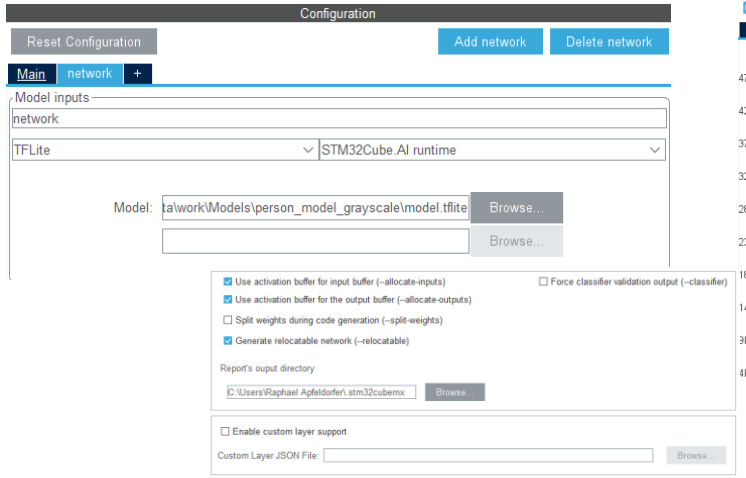


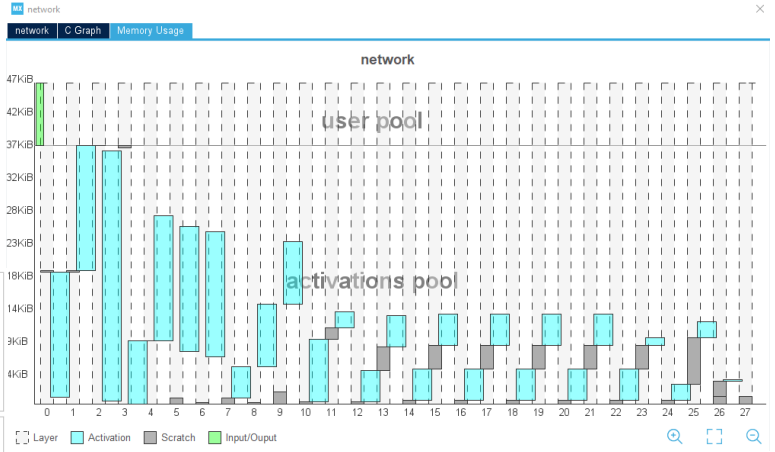
*	Part No	Reference	Marketing ...	Unit Price for 10...	Board	Package	Flash	RAM	IO	Freq.
☆	STM32F302RD	STM32F...	Active	2.846		LQFP64	384 kBytes	64 kBytes	51	72 MHz
☆	STM32F302RE	STM32F...	Active	3.24		LQFP64	512 kBytes	64 kBytes	51	72 MHz
☆	STM32F302VD	STM32F...	Active	3.216		UFBGA100	384 kBytes	64 kBytes	86	72 MHz
☆		STM32F...	Active	3.216		LQFP100	384 kBytes	64 kBytes	86	72 MHz
☆	STM32F302VE	STM32F...	Active	3.61		UFBGA100	512 kBytes	64 kBytes	86	72 MHz
☆		STM32F...	Active	3.61		LQFP100	512 kBytes	64 kBytes	86	72 MHz
☆	STM32F302ZD	STM32F...	Active	3.795		LQFP144	384 kBytes	64 kBytes	115	72 MHz
☆	STM32F302ZE	STM32F...	Active	4.188		LQFP144	512 kBytes	64 kBytes	115	72 MHz
☆	STM32F303RD	STM32F...	Active	3.09		LQFP64	384 kBytes	80 kBytes	51	72 MHz
☆	STM32F303RE	STM32F...	Active	3.483	NUCLEO-F303RE	LQFP64	512 kBytes	80 kBytes	51	72 MHz
☆	STM32F303VD	STM32F...	Active	3.517		UFBGA100	384 kBytes	80 kBytes	86	72 MHz

STM32Cube.AI user flow (2/3)

1. Select MCU
2. Optimize and validate
3. Generate project and integrate

- Model complexity and footprint analysis
- Fine tune memory allocation with optimizations and GUI
- Optimize system parameters and clock tree
- Extend model with your own customer layers





- Validate on desktop with your own dataset
- Validate on target and check inference time

Validate on desktop

Validate on target

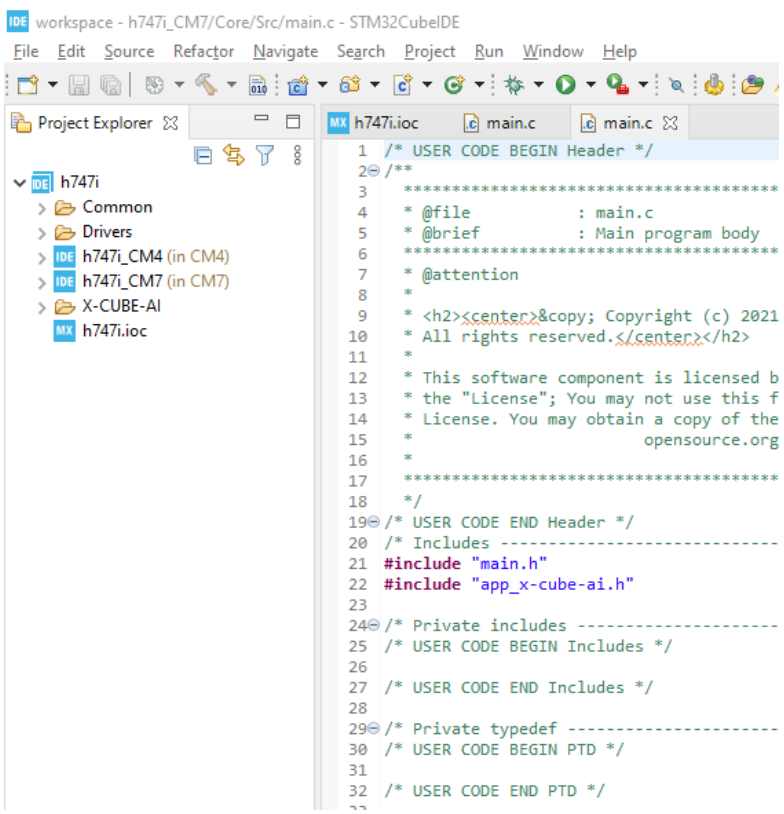
```
Results for 10 inference(s) - average per inference
 device               : 0x450 - STM32H743/53/50xx and STM32H745/55/47/57xx @400/200MHz fpu
 duration             : 35.475ms
 CPU cycles           : 14189943
 cycles/MACC          : 1.98
 c_nodes              : 28
c_id  m_id  desc                 oshape            fmt     ms         %
--------------------------------------------------------------------------------
0     0     Conv2D (0x103)       (1, 48, 48, 8)    uint8       3.453    9.7%
1     1     Conv2D (0x103)       (1, 48, 48, 8)    uint8       4.312   12.2%
2     2     Conv2D (0x103)       (1, 48, 48, 16)   uint8       2.104    5.9%
3     3     Conv2D (0x103)       (1, 24, 24, 16)   uint8       1.157    3.3%
4     4     Conv2D (0x103)       (1, 24, 24, 32)   uint8       1.250    3.5%
5     5     Conv2D (0x103)       (1, 24, 24, 32)   uint8       2.404    6.8%
6     6     Conv2D (0x103)       (1, 24, 24, 32)   uint8       1.911    5.4%
7     7     Conv2D (0x103)       (1, 12, 12, 32)   uint8       0.526    1.5%
8     8     Conv2D (0x103)       (1, 12, 12, 64)   uint8       0.884    2.5%
9     9     Conv2D (0x103)       (1, 12, 12, 64)   uint8       1.050    3.0%
10    10    Conv2D (0x103)       (1, 12, 12, 64)   uint8       1.544    4.4%
11    11    Conv2D (0x103)       (1, 6, 6, 64)     uint8       0.256    0.7%
12    12    Conv2D (0x103)       (1, 6, 6, 128)    uint8       0.793    2.2%
13    13    Conv2D (0x103)       (1, 6, 6, 128)    uint8       0.495    1.4%
14    14    Conv2D (0x103)       (1, 6, 6, 128)    uint8       1.453    4.1%
15    15    Conv2D (0x103)       (1, 6, 6, 128)    uint8       0.497    1.4%
16    16    Conv2D (0x103)       (1, 6, 6, 128)    uint8       1.454    4.1%
17    17    Conv2D (0x103)       (1, 6, 6, 128)    uint8       0.495    1.4%
18    18    Conv2D (0x103)       (1, 6, 6, 128)    uint8       1.453    4.1%
19    19    Conv2D (0x103)       (1, 6, 6, 128)    uint8       0.495    1.4%
20    20    Conv2D (0x103)       (1, 6, 6, 128)    uint8       1.454    4.1%
21    21    Conv2D (0x103)       (1, 6, 6, 128)    uint8       0.496    1.4%
22    22    Conv2D (0x103)       (1, 6, 6, 128)    uint8       1.453    4.1%
23    23    Conv2D (0x103)       (1, 3, 3, 128)    uint8       0.134    0.4%
24    24    Conv2D (0x103)       (1, 3, 3, 256)    uint8       0.917    2.6%
25    25    Conv2D (0x103)       (1, 3, 3, 256)    uint8       0.259    0.7%
26    27    Conv2dPool (0x109)   (1, 1, 1, 256)    uint8       2.763    7.8%
27    28    Conv2D (0x103)       (1, 1, 1, 3)      uint8       0.012    0.0%
--------------------------------------------------------------------------------
                                                               35.475 ms
```

STM32Cube.AI user flow (3/3)

1. Select MCU
2. Optimize and validate
3. Generate project and integrate

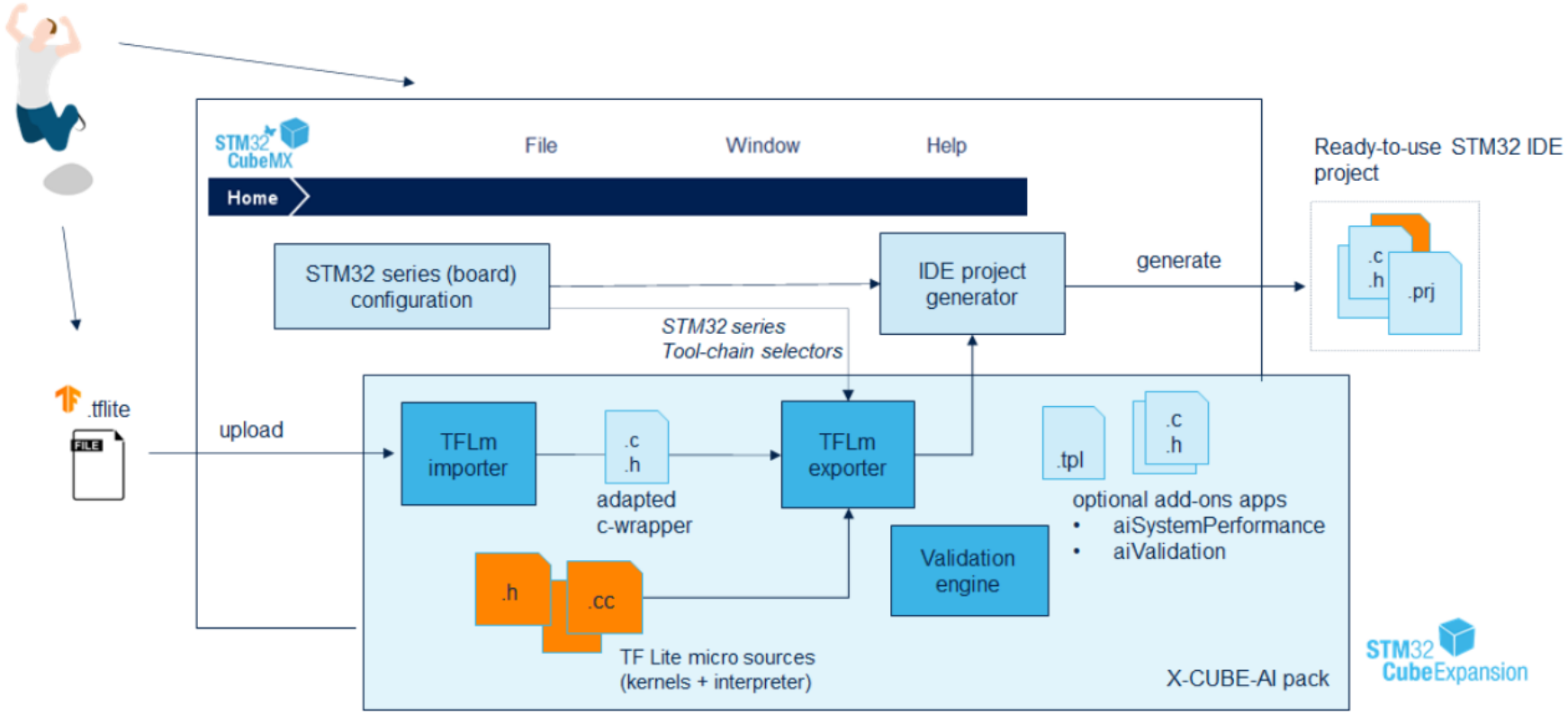
GENERATE CODE

- Generate Application Template
- Integrate with your application-specific code in your favorite IDE
- Perform system tests



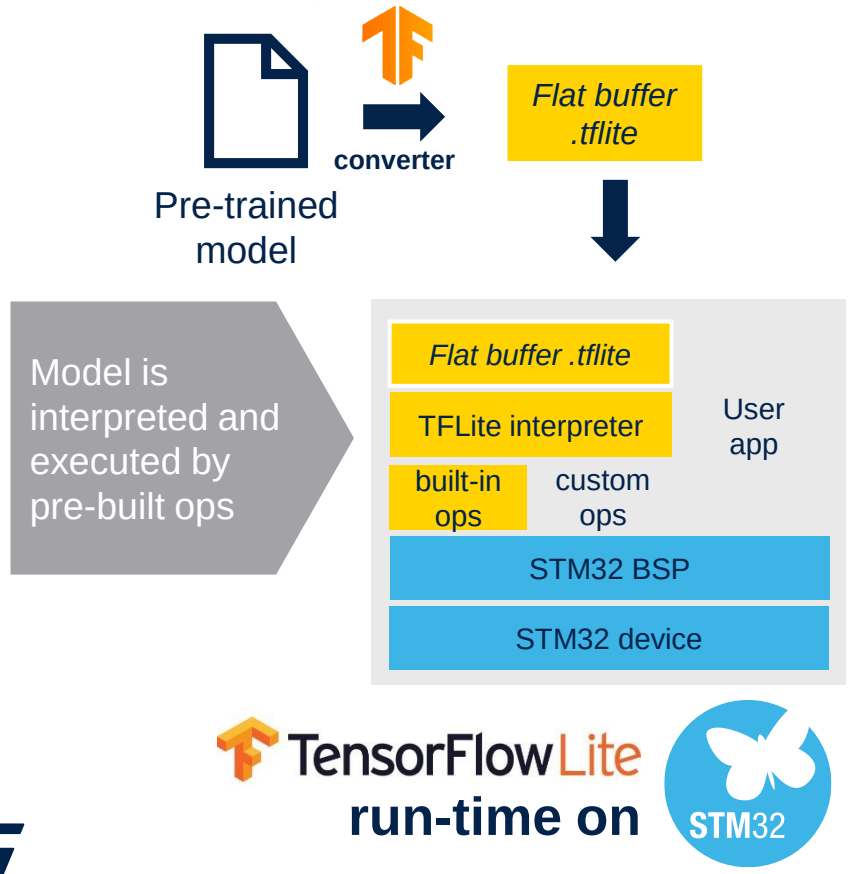
Possible conversion strategies: Network code generation and interpreter

- X-CUBE-AI Expansion Package integrates a specific path which allows to generate a ready-to-use STM32 IDE project embedding a TensorFlow Lite for Microcontrollers run-time (also called TFLm) and its associated TFLite model. This can be considered as an alternative of the default X-CUBE-AI solution to deploy a AI solution based on a TFLite model.

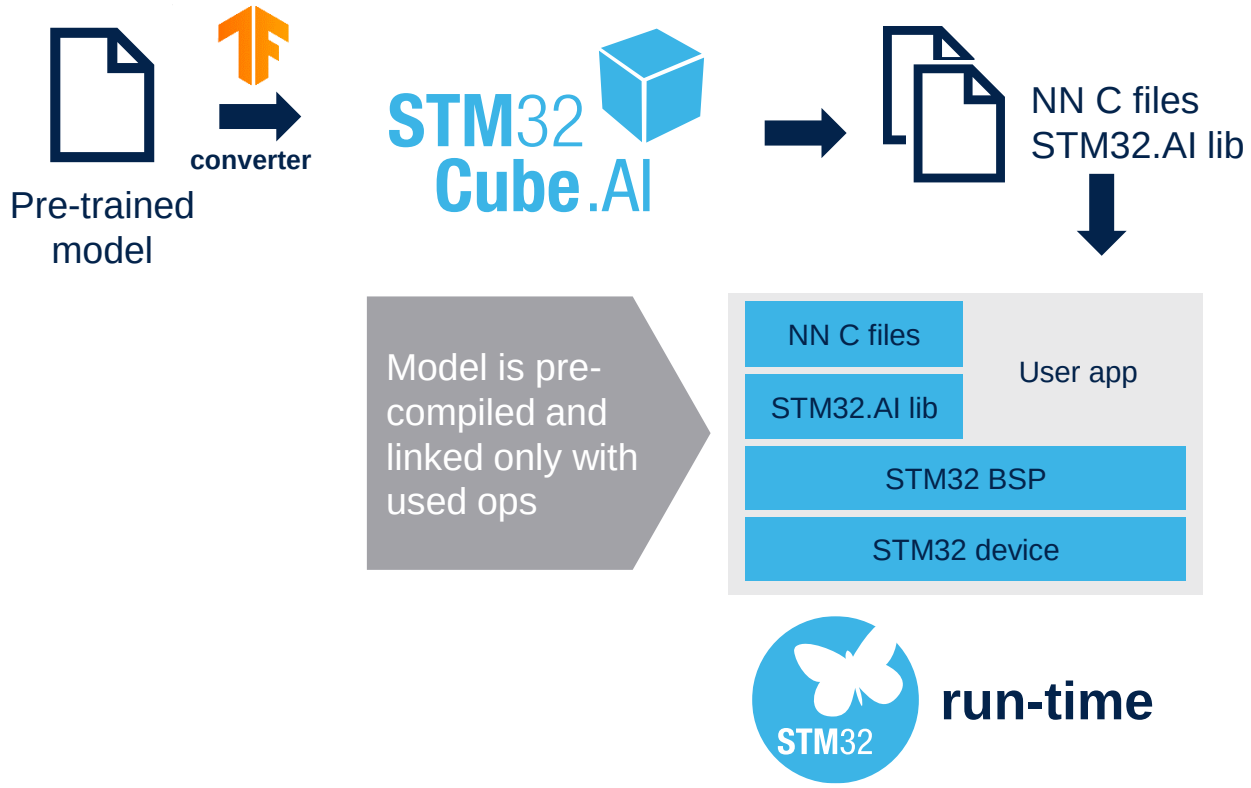


Possible conversion strategies: Network code generation and interpreter

More Flexible: TensorFlow Lite interpreter mode



More optimized: Optimized C code generated by STM32 Cube.AI



life.augmented

STM32Cube.AI vs Tensor Flow Lite for MCU

Results with STM32Cube.AI v7.2

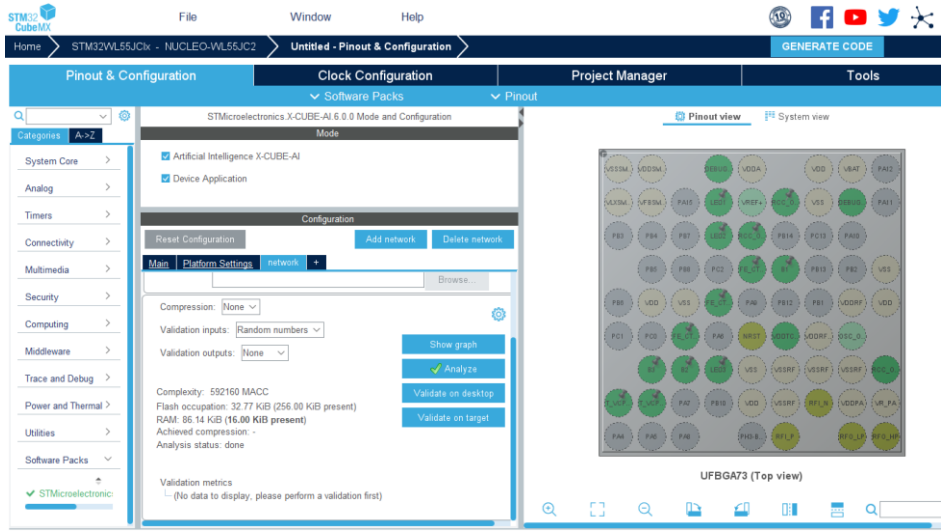
Model	Runtime	MCU and board	Inference time (ms)	Flash (KiB)	RAM (KiB)
Image Classification	X-CUBE-AI	STM32U585 NUCLEO-U575ZI-Q	148	142	50
Image Classification	TFLM		253	149	55
%	TFLM vs Cube.AI		+ 71	+ 5	+ 9

Model	Runtime	MCU and board	Inference time (ms)	Flash (KiB)	RAM (KiB)
Visual Wake Word	X-CUBE-AI	STM32H7A3 NUCLEO-H7A3ZI-Q	53	301	58
Visual Wake Word	TFLM		72	381	102
%	TFLM vs Cube.AI		+ 35	+ 27	+ 75

Two user interfaces of X-CUBE-AI

- In order to get a better user experience, X-CUBE-AI provides two user interfaces , GUI and CLI
- GUI can be used with cubeMX

GUI

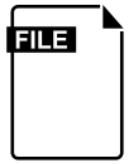


CLI

```
usage: stm32ai      -h] [--version] [--tools-version] [--model FILE]
                    [--verbosity [{0,1,2}]] [--type [keras|onnx|tflite]]
                    [--name STR] [--compression [1|4|8]] [--quantize [FILE]]
                    [--custom FILE] [--allocate-inputs] [--allocate-outputs]
                    [--workspace DIR] [--output DIR] [--split-weights]
                    [--binary] [--address ADDR] [--copy-weights-at ADDR]
                    [--relocatable] [--lib DIR] [--series STR] [--no-c-files]
                    [--batch-size INT] [--mode MODE] [--desc DESC]
                    [--valinput FILE [FILE ...]] [--valoutput FILE [FILE ...]]
                    [--range RANGE RANGE] [--full] [--save-csv] [--classifier]
                    [--no-check] [--no-exec-model] [--with-report]
                    analyze|generate|validate|quantize|supported-ops
```

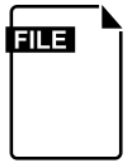
life.augmented

« stm32ai » main command(s)



Pre-trained model
(topology and weights)

Input/Output custom
data or random data
provided by the tool



Pre-trained model with
Quantization parameters

analyze

validate

generate

quantize

- Memory requirements *(ro/rw)*
- Processing requirements *(MACC)*
- Complexity per layer

- Metrics report, confusion matrix
- Accuracy classification
- Validation on target/desktop

- Generate C-files
- Generate binary file for training parameters

- Keras reshaped pre-trained model *(float)*
- Quantization parameters *(JSON file)*

life.augmented

Generate command

- The 'generate' command is used to generate the specialized network and data C-files. They are generated in the specified output directory.
- The '<name>.c/.h' files contain the topology of the C-model (C-struct definition of the tensors and the operators), including the embedded inference client API to use the generated model on the top of the optimized inference runtime library. '<name>_data.c/.h' files are by default a simple C-array with the data of the weight/bias tensors.

```
$ stm32ai generate -m <model_file_path> -o <output-directory-path>
...
Generated files (5)
--------------------------------------------------------------
 <output-directory-path>\<name>_config.h
 <output-directory-path>\<name>.c
 <output-directory-path>\<name>_data.c
 <output-directory-path>\<name>.h
 <output-directory-path>\<name>_data.h

Creating report file <output-directory-path>\network_generate_report.txt
...
```

Analyze command

- The 'analyze' command is the primary command to import, to parse, to check and to render an uploaded pre-trained model. Detailed report provides the main system metrics to know if the generated code can be deployed on a STM32 device. It includes also rendering information by layer or/and operator.

```
$ stm32ai analyze -m ds_cnn.h5
Neural Network Tools for STM32AI v1.4.1 (STM.ai v6.0.0)

 Exec/report summary (analyze)
 ------------------------------------------------------------------------------
 model file         : <model-directory-path>\ds_cnn.h5
 type               : keras
 c_name             : network
 compression        : None
 quantize           : None
 workspace dir      : <workspace-directory-path>
 output dir         : <output-directory-path>

 model_name         : ds_cnn
 model_hash         : cea66c686b2a1529d82848dc1fd0d873
 input              : input_0 [490 items, 1.91 KiB, ai_float, FLOAT32, (49, 10, 1)]
 inputs (total)     : 1.91 KiB
 output             : dense_1_nl [12 items, 48 B, ai_float, FLOAT32, (1, 1, 12)]
 outputs (total)    : 48 B
 params #           : 40,140 items (156.80 KiB)
 macc               : 4,833,536
 weights (ro)       : 159,536 B (155.80 KiB) -1,024(-0.6%)
 activations (rw)   : 57,600 B (56.25 KiB)
 ram (total)        : 59,608 B (58.21 KiB) = 57,600 + 1,960 + 48
```

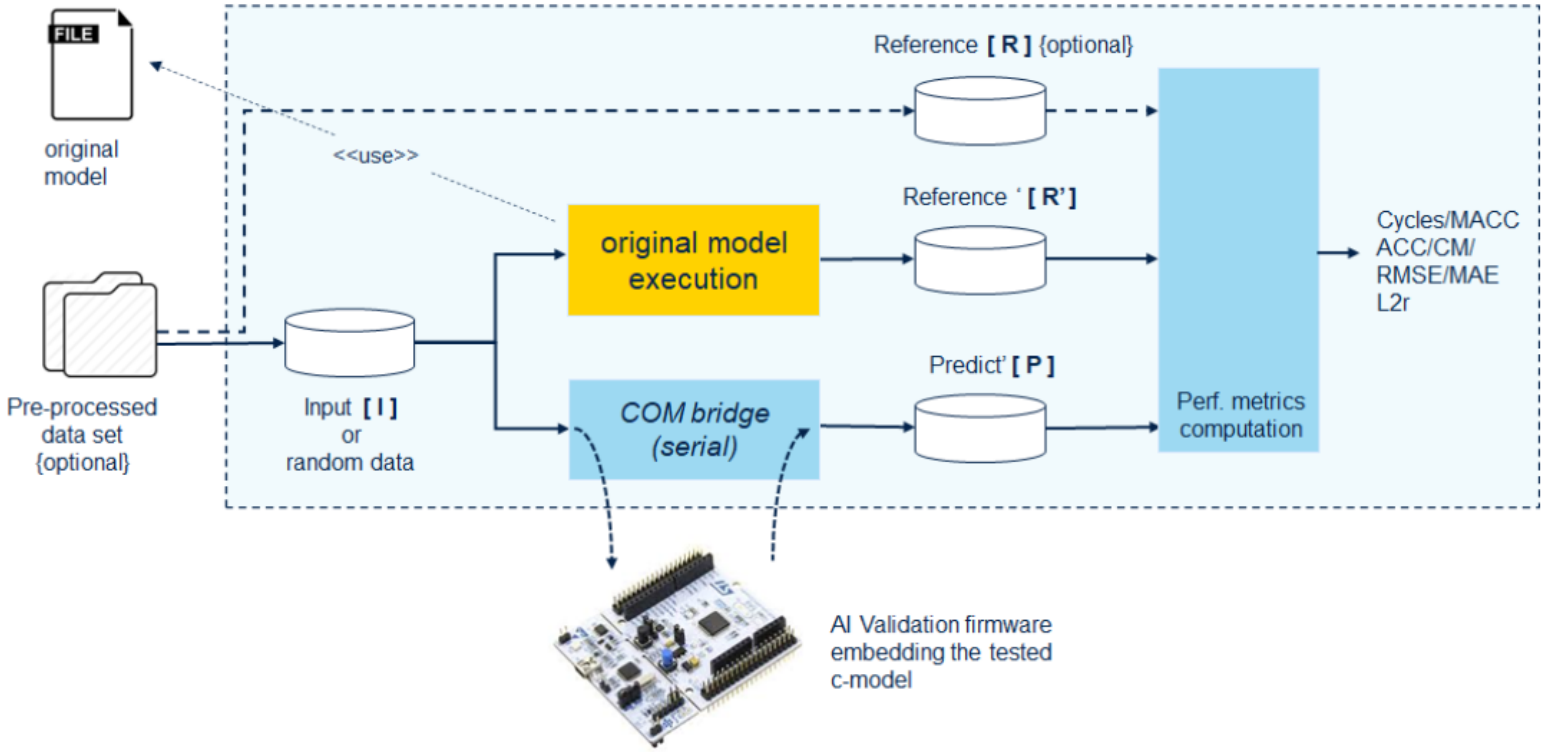
Validate command

- The 'validate' command allows to import, to render and to validate the generated C-files.

```
$ stm32ai validate -m <model_f32p_file_path> --full
...
 Complexity report per layer - macc=4,013 weights=15,560 act=192 ram_io=416
 ------------------------------------------------------------------------------------------------------
 id   name           c_macc                    c_rom                     c_id   c_dur    l2r
 ------------------------------------------------------------------------------------------------------
 0    dense_1        |||||||||||||||  82.2%   |||||||||||||||  84.8%   [0]     71.4%   5.62010030e-08
 1    activation_1   |                 0.8%    |                 0.0%    [1]      7.1%   5.57235715e-08
 2    dense_2        |||              12.7%    |||              13.1%    [2]      7.1%   8.20674515e-08
 3    activation_2   |                 0.4%    |                 0.0%    [3]      0.0%   8.00048383e-08
 4    dense_3        |                 2.0%    |                 2.1%    [4]      0.0%   1.32168850e-07
 5    activation_3   |                 1.9%    |                 0.0%    [5]     14.3%   3.99808840e-07 *
...
```

Validation

- the different metrics (and associated computing flow) which are used to evaluate the performance of the generated C-files (or C-model). Proposed metrics should be considered as the generic indicators which allows to compare numerically the predictions of the c-model against the predictions of the original model.




```
Evaluation report (summary)
------------------------------------------------------------
Mode                 acc      rmse      mae       l2r
------------------------------------------------------------
x86 C-model #1       92.68%   0.053623  0.005785  0.340042
original model #1    92.68%   0.053623  0.005785  0.340042
X-cross #1           100.00%  0.000000  0.000000  0.000000
------------------------------------------------------------
```

life.augmented

Don't go alone

A network of companies is there to support you

Trust our **authorized partners** to ensure the success of your project. Learn more at st.com/stm32ai

Would you like to discuss a co-development partnership for ML/AI projects? Contact us at edge.ai@st.com

life.augmented

Our technology starts with You

Find out more at www.st.com/stm32ai

life.augmented