Demo CubeMX-AI:
A CIFAR10 classification

Alessio Burrello,
alessio.burrello@unibo.it

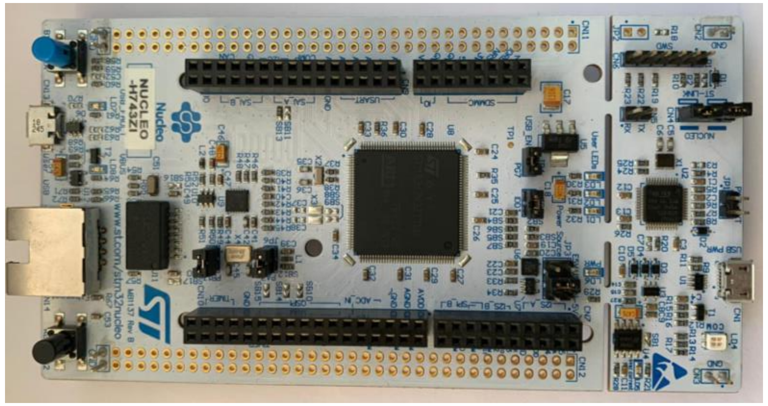
Credits: Marcello Zanghieri,
Enrico Tabanelli

Summary

- Creation of the python model and production of the saved model (.h5)
- Creation of the CubeMX project and validation on NUCLEO-H743ZI
- Creation of the application for network inference

Material

- NUCLEO-H743ZI



- Cable to connect

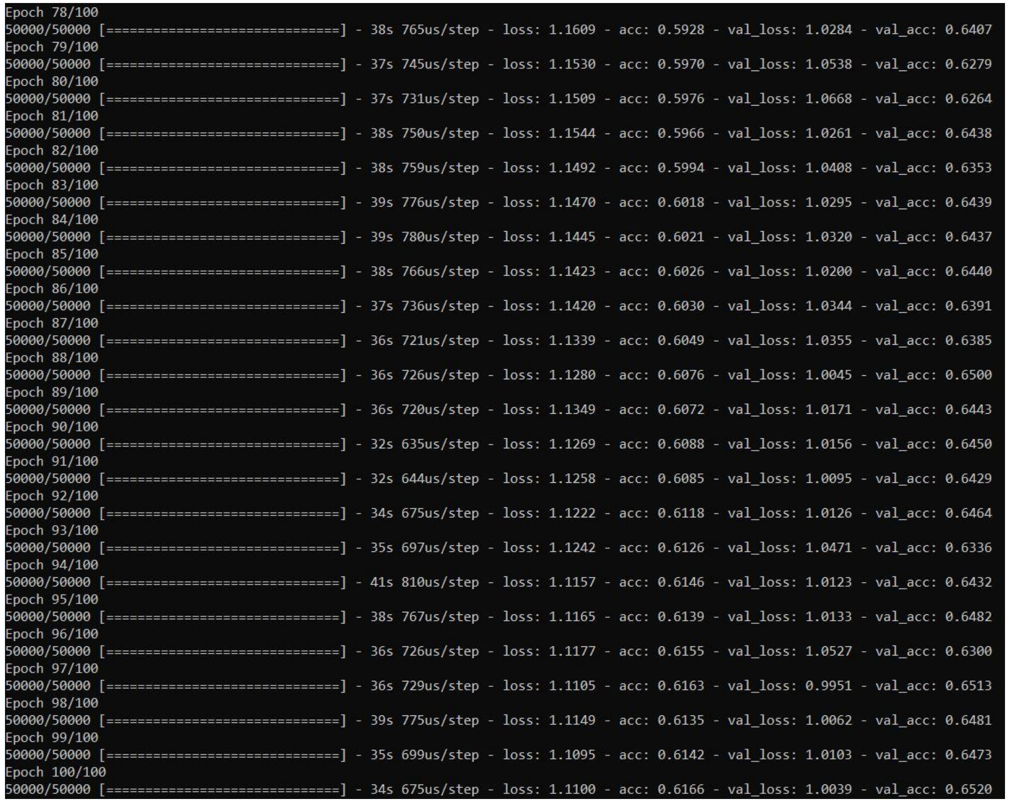
- PC with Windows + VM with Linux
  - Python 3.x
  - STMCubeMX-AI
  - STM32CubeIDE

Summary

- **Creation of the python network and production of the saved model (.h5)**
- Creation of the CubeMX project and validation on NUCLEO-H743ZI
- Creation of the application for network inference

Python network generation: keras

Training of the network



```
Epoch 78/100
50000/50000 [==============================] - 38s 765us/step - loss: 1.1609 - acc: 0.5928 - val_loss: 1.0284 - val_acc: 0.6407
Epoch 79/100
50000/50000 [==============================] - 37s 745us/step - loss: 1.1530 - acc: 0.5970 - val_loss: 1.0538 - val_acc: 0.6279
Epoch 80/100
50000/50000 [==============================] - 37s 731us/step - loss: 1.1509 - acc: 0.5976 - val_loss: 1.0668 - val_acc: 0.6264
Epoch 81/100
50000/50000 [==============================] - 38s 750us/step - loss: 1.1544 - acc: 0.5966 - val_loss: 1.0261 - val_acc: 0.6438
Epoch 82/100
50000/50000 [==============================] - 38s 759us/step - loss: 1.1492 - acc: 0.5994 - val_loss: 1.0408 - val_acc: 0.6353
Epoch 83/100
50000/50000 [==============================] - 39s 776us/step - loss: 1.1470 - acc: 0.6018 - val_loss: 1.0295 - val_acc: 0.6439
Epoch 84/100
50000/50000 [==============================] - 39s 780us/step - loss: 1.1445 - acc: 0.6021 - val_loss: 1.0320 - val_acc: 0.6437
Epoch 85/100
50000/50000 [==============================] - 38s 766us/step - loss: 1.1423 - acc: 0.6026 - val_loss: 1.0200 - val_acc: 0.6440
Epoch 86/100
50000/50000 [==============================] - 37s 736us/step - loss: 1.1420 - acc: 0.6030 - val_loss: 1.0344 - val_acc: 0.6391
Epoch 87/100
50000/50000 [==============================] - 36s 721us/step - loss: 1.1339 - acc: 0.6049 - val_loss: 1.0355 - val_acc: 0.6385
Epoch 88/100
50000/50000 [==============================] - 36s 726us/step - loss: 1.1280 - acc: 0.6076 - val_loss: 1.0045 - val_acc: 0.6500
Epoch 89/100
50000/50000 [==============================] - 36s 720us/step - loss: 1.1349 - acc: 0.6072 - val_loss: 1.0171 - val_acc: 0.6443
Epoch 90/100
50000/50000 [==============================] - 32s 635us/step - loss: 1.1269 - acc: 0.6088 - val_loss: 1.0156 - val_acc: 0.6450
Epoch 91/100
50000/50000 [==============================] - 32s 644us/step - loss: 1.1258 - acc: 0.6085 - val_loss: 1.0095 - val_acc: 0.6429
Epoch 92/100
50000/50000 [==============================] - 34s 675us/step - loss: 1.1222 - acc: 0.6118 - val_loss: 1.0126 - val_acc: 0.6464
Epoch 93/100
50000/50000 [==============================] - 35s 697us/step - loss: 1.1242 - acc: 0.6126 - val_loss: 1.0471 - val_acc: 0.6336
Epoch 94/100
50000/50000 [==============================] - 41s 810us/step - loss: 1.1157 - acc: 0.6146 - val_loss: 1.0123 - val_acc: 0.6432
Epoch 95/100
50000/50000 [==============================] - 38s 767us/step - loss: 1.1165 - acc: 0.6139 - val_loss: 1.0133 - val_acc: 0.6482
Epoch 96/100
50000/50000 [==============================] - 36s 726us/step - loss: 1.1177 - acc: 0.6155 - val_loss: 1.0527 - val_acc: 0.6300
Epoch 97/100
50000/50000 [==============================] - 36s 729us/step - loss: 1.1105 - acc: 0.6163 - val_loss: 0.9951 - val_acc: 0.6513
Epoch 98/100
50000/50000 [==============================] - 39s 775us/step - loss: 1.1149 - acc: 0.6135 - val_loss: 1.0062 - val_acc: 0.6481
Epoch 99/100
50000/50000 [==============================] - 35s 699us/step - loss: 1.1095 - acc: 0.6142 - val_loss: 1.0103 - val_acc: 0.6473
Epoch 100/100
50000/50000 [==============================] - 34s 675us/step - loss: 1.1100 - acc: 0.6166 - val_loss: 1.0039 - val_acc: 0.6520
```

How to train

- Launch CIFAR10 network example, that you find in the folder
- Change the following parameters, to generate more models:

  batch_size = 32
  num_classes = 10
  epochs = 200
  data_augmentation = True
  num_predictions = 20
  save_dir = os.path.join(os.getcwd(), 'saved_models’)
  model_name = 'keras_cifar10_trained_model_200_epochs.h5'

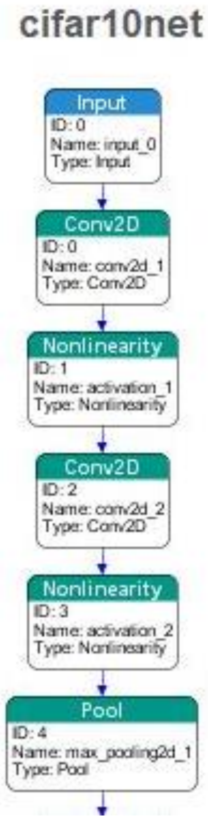
Python network generation: keras

Generated model:

- .h5 file, with topology + weights
- a CNN network with conv2d, poolings, Relu, Dropout and Dense Layers

Full network contains:

- 3 blocks identical to the presented one;
- 2 final Fully connected layers

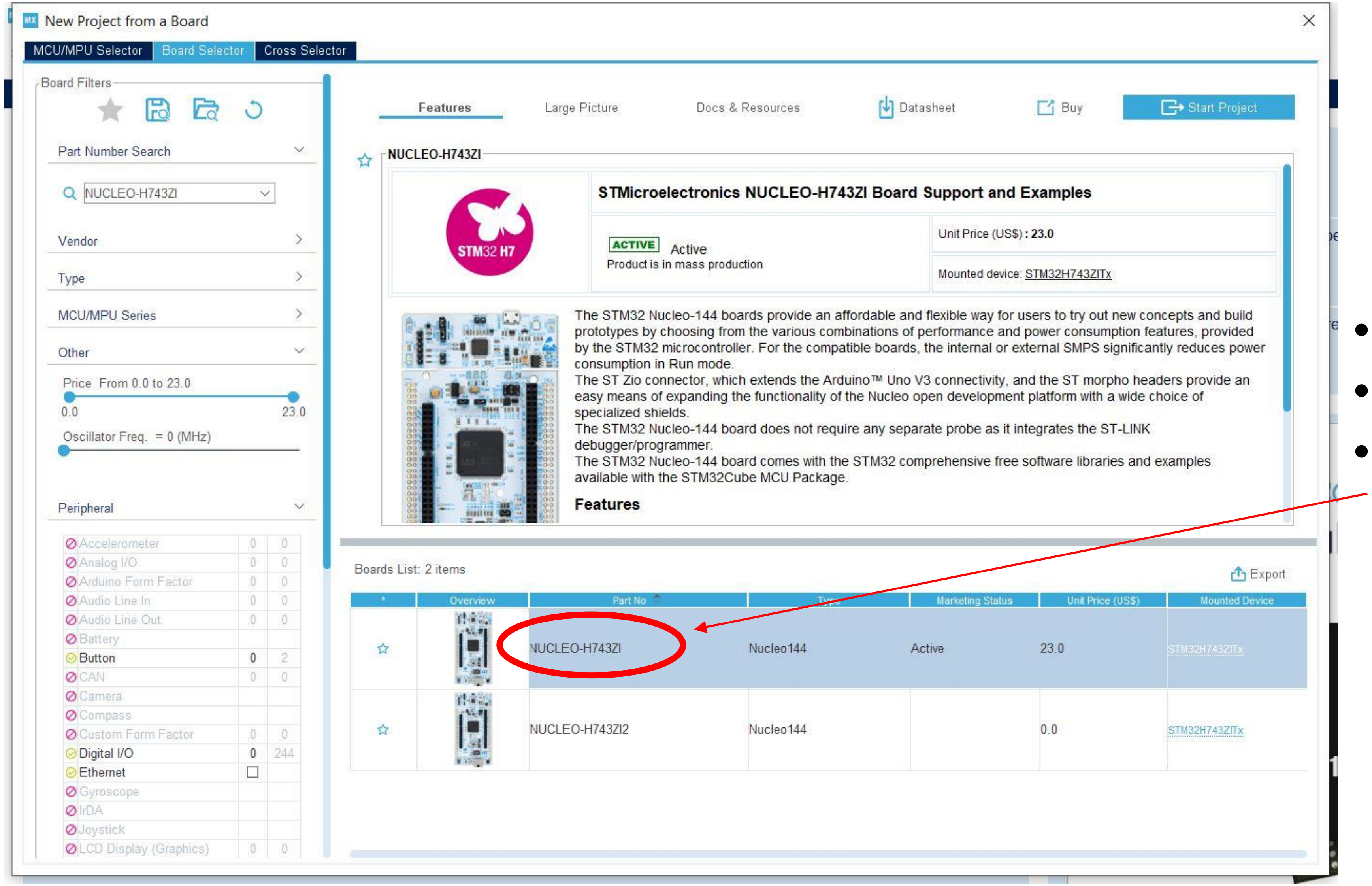


Block 1 as visualized from CubeMX-AI

Summary

- Creation of the python network and production of the saved model (.h5)
- **Creation of the CubeMX project and validation on NUCLEO-H743ZI**
- Creation of the application for network inference

STM32CubeMX: New project



New Project from a Board

Part Number Search: NUCLEO-H743ZI

Price From 0.0 to 23.0

0.0 23.0

Oscillator Freq. = 0 (MHz)

Peripheral

Accelerometer	0	0
Analog I/O	0	0
Arduino Form Factor	0	0
Audio Line In	0	0
Audio Line Out	0	0
Battery		
Button	0	2
CAN	0	0
Camera		
Compass		
Custom Form Factor	0	0
Digital I/O	0	244
Ethernet	□	
Gyroscope		
IrDA		
Joystick		
LCD Display (Graphics)	0	0

NUCLEO-H743ZI

STMicroelectronics NUCLEO-H743ZI Board Support and Examples

STM32 H7

ACTIVE Active
Product is in mass production

Unit Price (US$) : 23.0

Mounted device: STM32H743ZITx

The STM32 Nucleo-144 boards provide an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features, provided by the STM32 microcontroller. For the compatible boards, the internal or external SMPS significantly reduces power consumption in Run mode.
The ST Zio connector, which extends the Arduino™ Uno V3 connectivity, and the ST morpho headers provide an easy means of expanding the functionality of the Nucleo open development platform with a wide choice of specialized shields.
The STM32 Nucleo-144 board does not require any separate probe as it integrates the ST-LINK debugger/programmer.
The STM32 Nucleo-144 board comes with the STM32 comprehensive free software libraries and examples available with the STM32Cube MCU Package.

Features

Boards List: 2 items

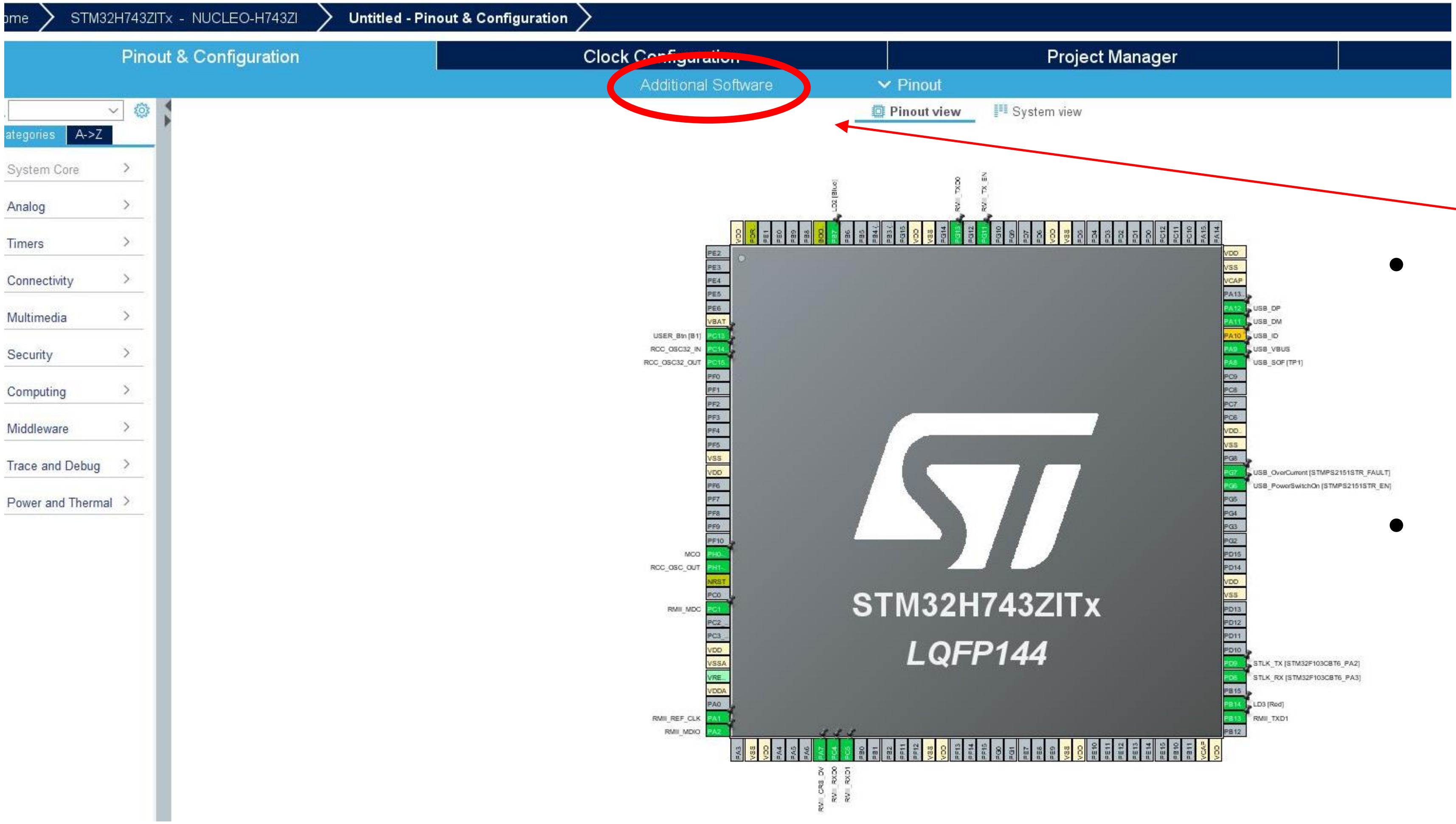
Export

*	Overview	Part No	Type	Marketing Status	Unit Price (US$)	Mounted Device
☆		NUCLEO-H743ZI	Nucleo144	Active	23.0	STM32H743ZITx
☆		NUCLEO-H743ZI2	Nucleo144		0.0	STM32H743ZITx

- Create a new project
- Select **Board Selector**
- Select **NUCLEO-H743ZI**

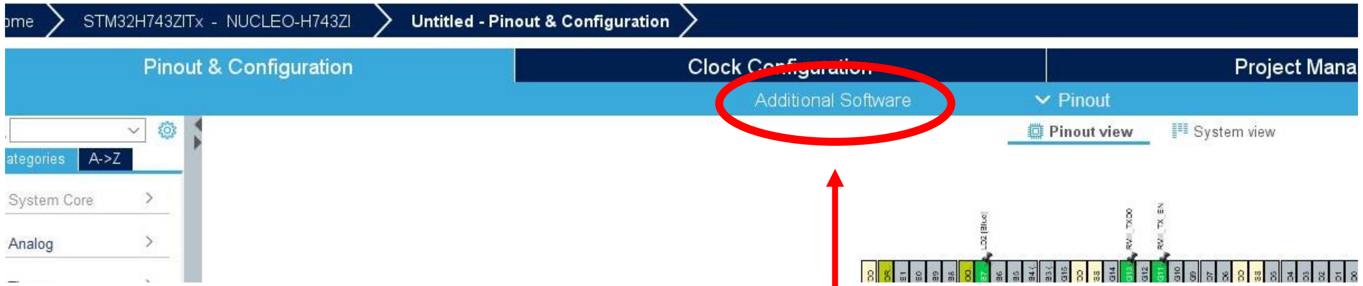
Select a different board if you have a different one.

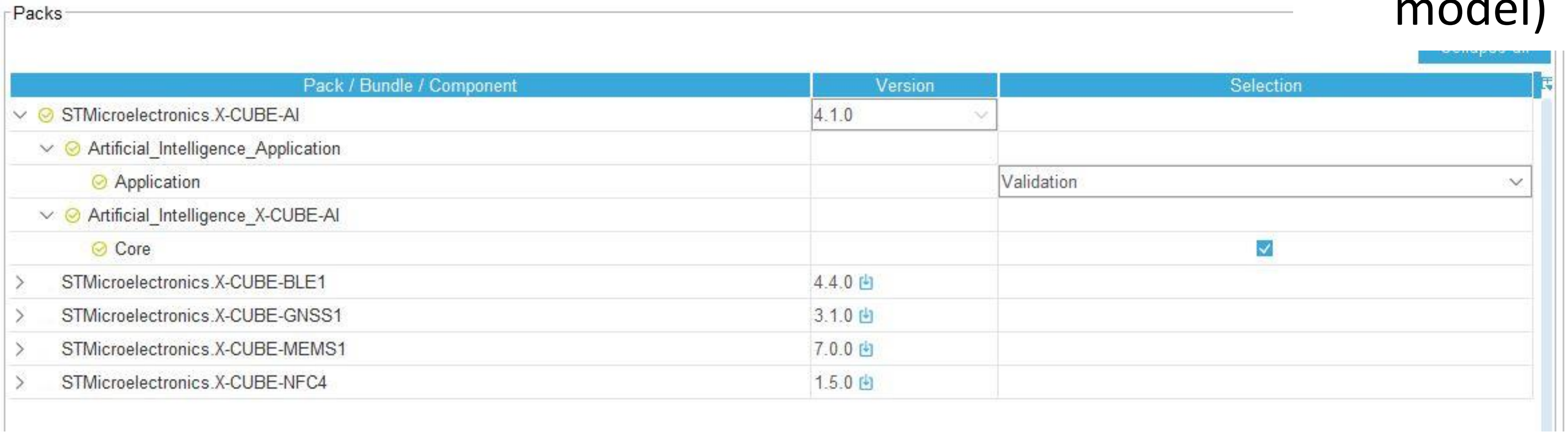
STM32CubeMX: add AI



- Use the *Additional software* to add a validation application of a Neural network.
- Validate the execution of the NN on the NUCLEO-H7.

STM32CubeMX: add AI

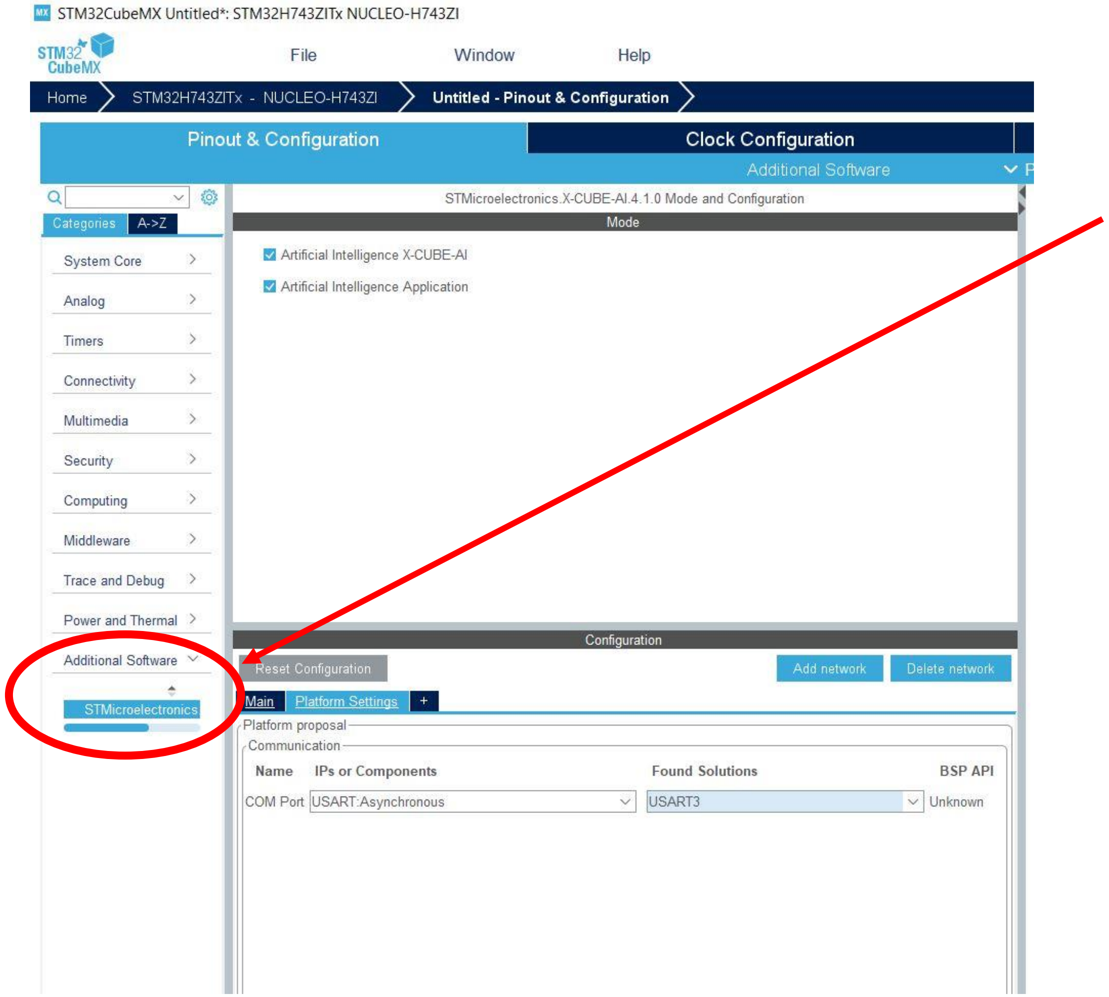




Pack / Bundle / Component	Version	Selection
STMicroelectronics.X-CUBE-AI	4.1.0	
Artificial_Intelligence_Application		
Application		Validation
Artificial_Intelligence_X-CUBE-AI		
Core		☑
STMicroelectronics.X-CUBE-BLE1	4.4.0	
STMicroelectronics.X-CUBE-GNSS1	3.1.0	
STMicroelectronics.X-CUBE-MEMS1	7.0.0	
STMicroelectronics.X-CUBE-NFC4	1.5.0	

- Select *Validation* to validate your application (check memory and accuracy of the result → MSE compared to PC model)

STM32CubeMX-AI: Configuration of the network



STM32CubeMX Untitled*: STM32H743ZITx NUCLEO-H743ZI

Home > STM32H743ZITx - NUCLEO-H743ZI > Untitled - Pinout & Configuration

STMicroelectronics.X-CUBE-AI.4.1.0 Mode and Configuration

☑ Artificial Intelligence X-CUBE-AI

☑ Artificial Intelligence Application

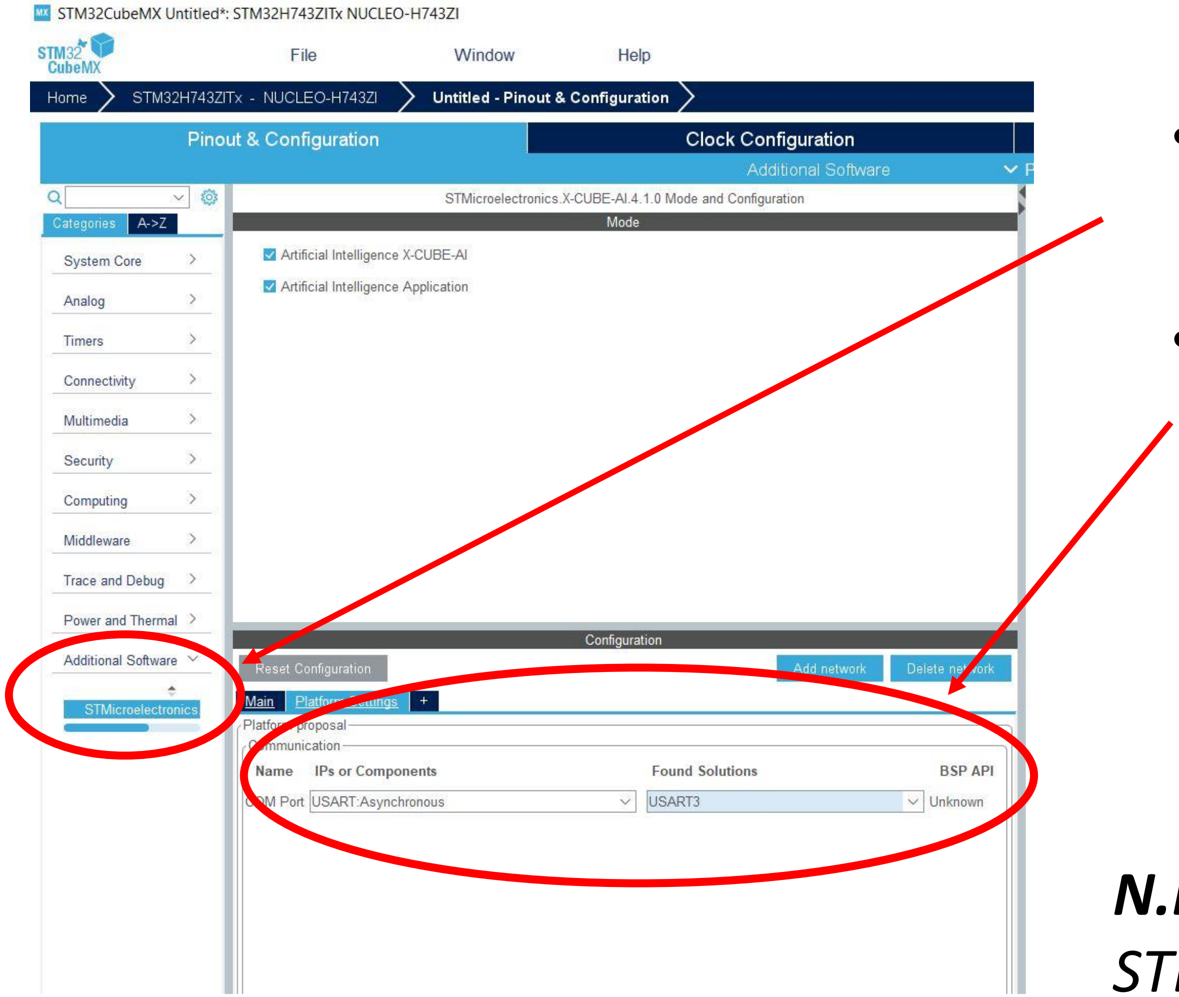
Platform proposal

Communication

Name	IPs or Components	Found Solutions	BSP API
COM Port	USART:Asynchronous	USART3	Unknown

- Additional software configured

STM32CubeMX-AI: Configuration of the network



STM32CubeMX Untitled*: STM32H743ZITx NUCLEO-H743ZI

STMicroelectronics.X-CUBE-AI.4.1.0 Mode and Configuration

☑ Artificial Intelligence X-CUBE-AI

☑ Artificial Intelligence Application

Platform proposal

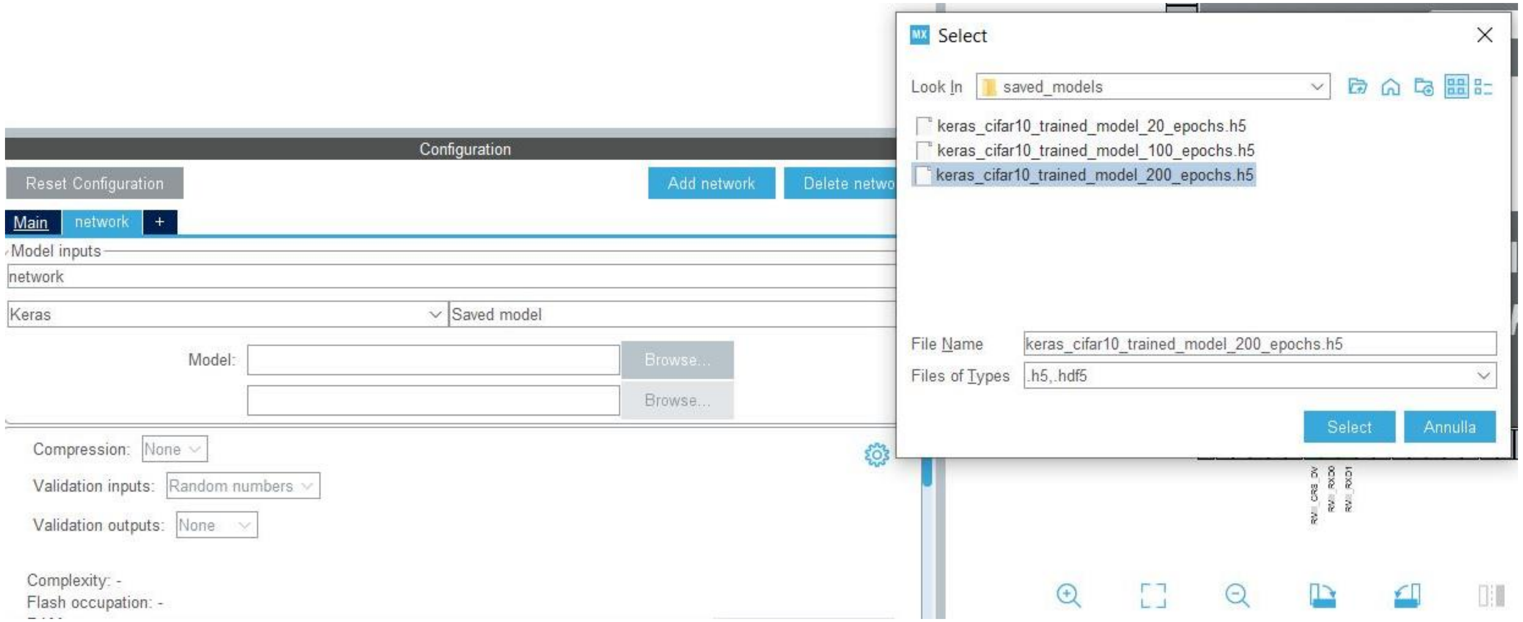
Communication

Name	IPs or Components	Found Solutions	BSP API
COM Port	USART:Asynchronous	USART3	Unknown

- Additional software configured
- Configure the USART3 for validation in Platform settings.
  - If not automatically activated, remember to activate USART3 in *Connectivity*

***N.B.** USART3 is the one connected to the STLINK for NUCLEO-H7. Check the right one on your board.*

STM32CubeMX-AI: Add a network



Model inputs: network

Keras | Saved model

Model:

Compression: None

Validation inputs: Random numbers

Validation outputs: None

Complexity: -

Flash occupation: -

keras_cifar10_trained_model_20_epochs.h5

keras_cifar10_trained_model_100_epochs.h5

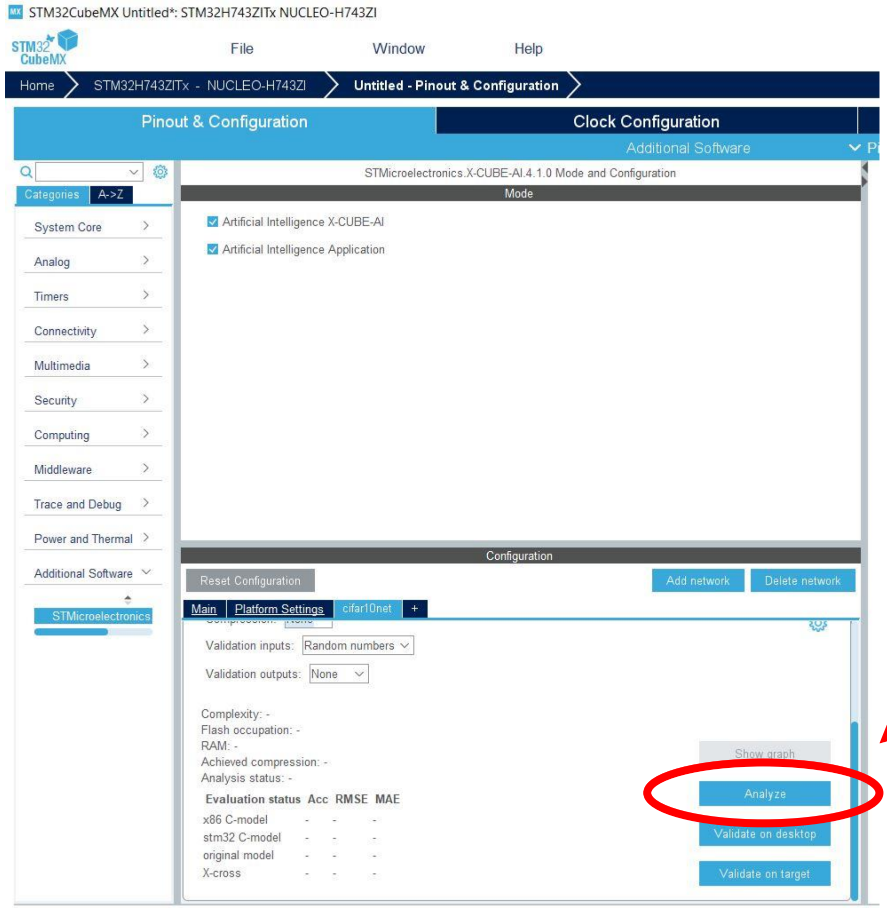
keras_cifar10_trained_model_200_epochs.h5

File Name: keras_cifar10_trained_model_200_epochs.h5

Files of Types: .h5,.hdf5

- Select *keras*, *saved model*.
- Change name of Model input to ***cifar10net***
- Import a network from saved models in github (or a generic .h5 network)

STM32CubeMX-AI: actions



STMicroelectronics.X-CUBE-AI.4.1.0 Mode and Configuration

☑ Artificial Intelligence X-CUBE-AI

☑ Artificial Intelligence Application

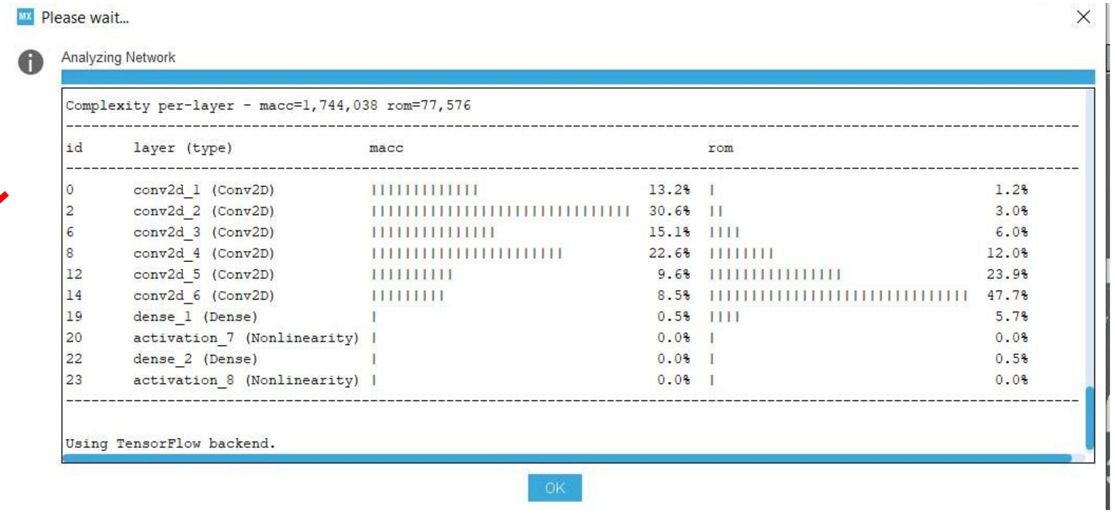
Validation inputs: Random numbers

Validation outputs: None

Complexity: -
Flash occupation: -
RAM: -
Achieved compression: -
Analysis status: -

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

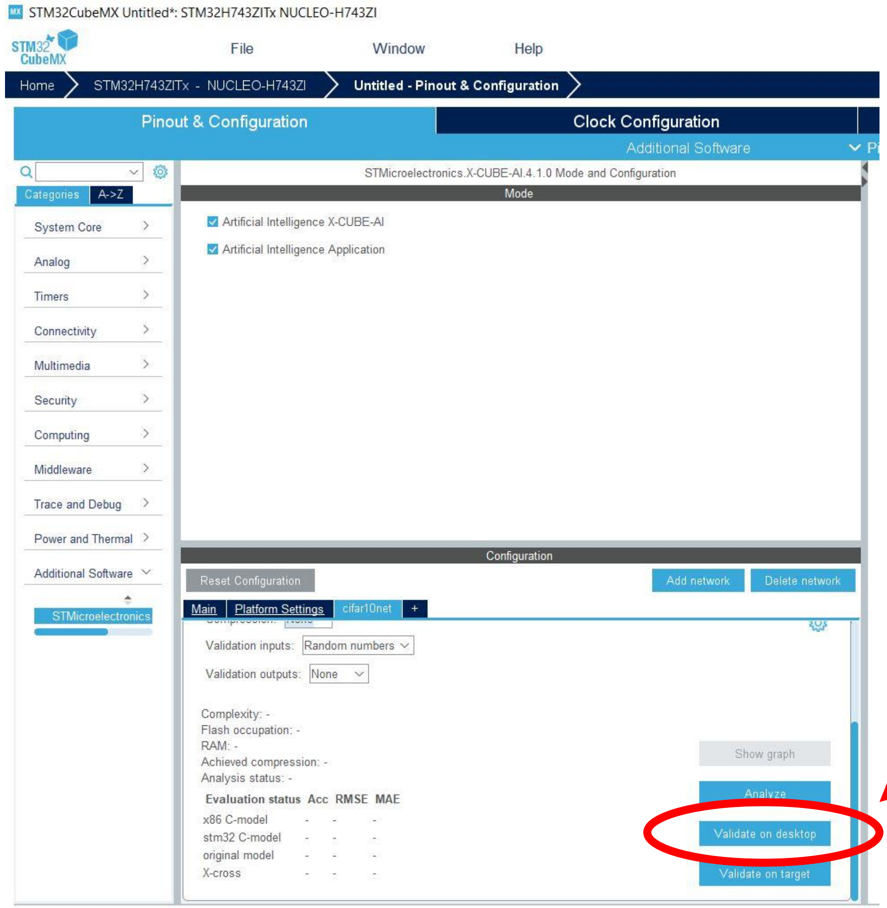
Analyze. Check the correctness of the network. You can choose a level of compression (None, 4, 8)



Analyzing Network

```
Complexity per-layer - macc=1,744,038 rom=77,576
------------------------------------------------------------------------------------------------------------------------
id      layer (type)                macc                                          rom
------------------------------------------------------------------------------------------------------------------------
0       conv2d_1 (Conv2D)           |||||||||||||                      13.2%  |                                  1.2%
2       conv2d_2 (Conv2D)           |||||||||||||||||||||||||||||||    30.6%  ||                                 3.0%
6       conv2d_3 (Conv2D)           |||||||||||||||                    15.1%  ||||                               6.0%
8       conv2d_4 (Conv2D)           |||||||||||||||||||||||            22.6%  ||||||||                          12.0%
12      conv2d_5 (Conv2D)           ||||||||||                          9.6%  ||||||||||||||||                  23.9%
14      conv2d_6 (Conv2D)           |||||||||                           8.5%  |||||||||||||||||||||||||||||||   47.7%
19      dense_1 (Dense)             |                                   0.5%  ||||                               5.7%
20      activation_7 (Nonlinearity) |                                   0.0%  |                                  0.0%
22      dense_2 (Dense)             |                                   0.0%  |                                  0.5%
23      activation_8 (Nonlinearity) |                                   0.0%  |                                  0.0%
------------------------------------------------------------------------------------------------------------------------

Using TensorFlow backend.
```

STM32CubeMX-AI: actions



STMicroelectronics.X-CUBE-AI.4.1.0 Mode and Configuration

Mode

☑ Artificial Intelligence X-CUBE-AI

☑ Artificial Intelligence Application

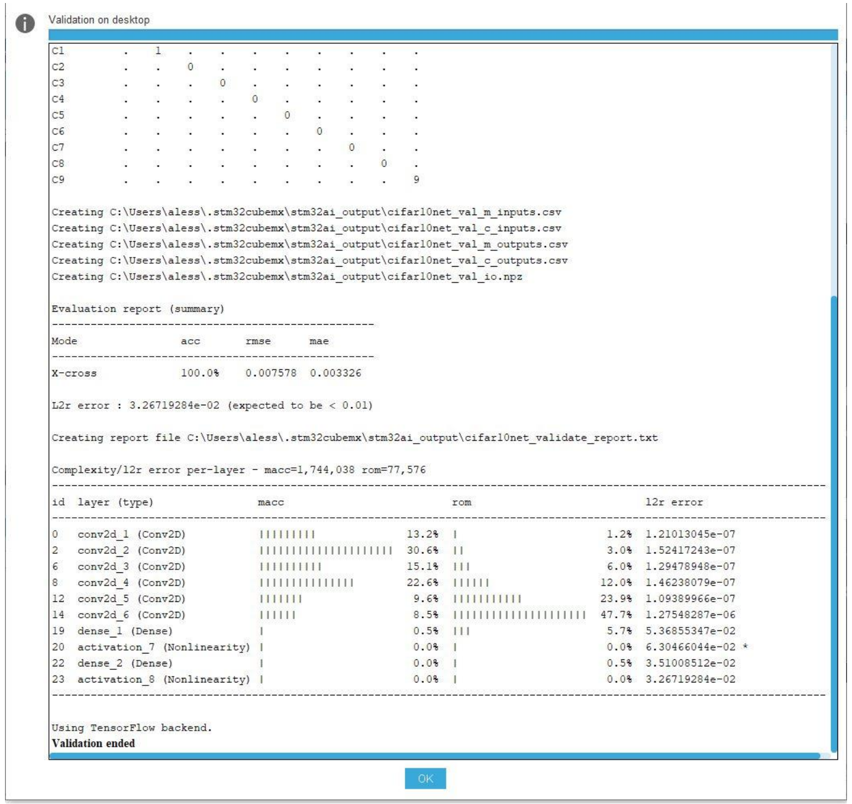
Validation inputs: Random numbers

Validation outputs: None

Complexity: -
Flash occupation: -
RAM: -
Achieved compression: -
Analysis status: -

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

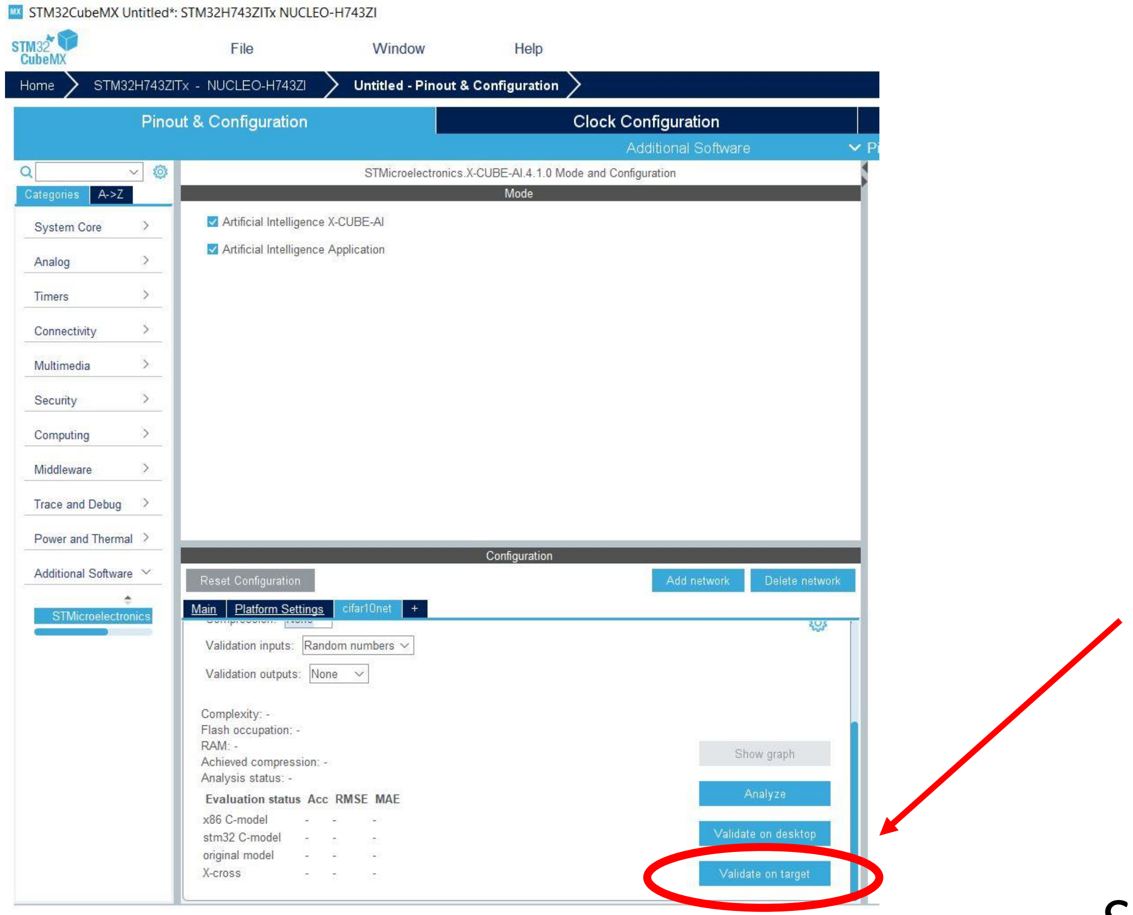
Validate on desktop



Validation on desktop

```
C1          .    1    .    .    .    .    .    .    .    .
C2          .    .    0    .    .    .    .    .    .    .
C3          .    .    .    0    .    .    .    .    .    .
C4          .    .    .    .    0    .    .    .    .    .
C5          .    .    .    .    .    0    .    .    .    .
C6          .    .    .    .    .    .    0    .    .    .
C7          .    .    .    .    .    .    .    0    .    .
C8          .    .    .    .    .    .    .    .    0    .
C9          .    .    .    .    .    .    .    .    .    9

Creating C:\Users\aless\.stm32cubemx\stm32ai_output\cifar10net_val_m_inputs.csv
Creating C:\Users\aless\.stm32cubemx\stm32ai_output\cifar10net_val_c_inputs.csv
Creating C:\Users\aless\.stm32cubemx\stm32ai_output\cifar10net_val_m_outputs.csv
Creating C:\Users\aless\.stm32cubemx\stm32ai_output\cifar10net_val_c_outputs.csv
Creating C:\Users\aless\.stm32cubemx\stm32ai_output\cifar10net_val_io.npz

Evaluation report (summary)
--------------------------------------------------
Mode                acc       rmse      mae
--------------------------------------------------
X-cross             100.0%    0.007578  0.003326

L2r error : 3.26719284e-02 (expected to be < 0.01)

Creating report file C:\Users\aless\.stm32cubemx\stm32ai_output\cifar10net_validate_report.txt

Complexity/l2r error per-layer - macc=1,744,038 rom=77,576
------------------------------------------------------------------------------------------------------------------------
id  layer (type)               macc                           rom                                l2r error
------------------------------------------------------------------------------------------------------------------------
0   conv2d_1 (Conv2D)          |||||||||              13.2%   |                           1.2%   1.21013045e-07
2   conv2d_2 (Conv2D)          |||||||||||||||||||||  30.6%   ||                          3.0%   1.52417243e-07
6   conv2d_3 (Conv2D)          ||||||||||             15.1%   |||                         6.0%   1.29478948e-07
8   conv2d_4 (Conv2D)          |||||||||||||||        22.6%   ||||||                     12.0%   1.46238079e-07
12  conv2d_5 (Conv2D)          |||||||                 9.6%   |||||||||||                23.9%   1.09389966e-07
14  conv2d_6 (Conv2D)          ||||||                  8.5%   |||||||||||||||||||||      47.7%   1.27548287e-06
19  dense_1 (Dense)            |                       0.5%   |||                         5.7%   5.36855347e-02
20  activation_7 (Nonlinearity) |                      0.0%   |                           0.0%   6.30466044e-02 *
22  dense_2 (Dense)            |                       0.0%   |                           0.5%   3.51008512e-02
23  activation_8 (Nonlinearity) |                      0.0%   |                           0.0%   3.26719284e-02
------------------------------------------------------------------------------------------------------------------------

Using TensorFlow backend.
Validation ended
```

STM32CubeMX-AI: actions



☑ Artificial Intelligence X-CUBE-AI

☑ Artificial Intelligence Application

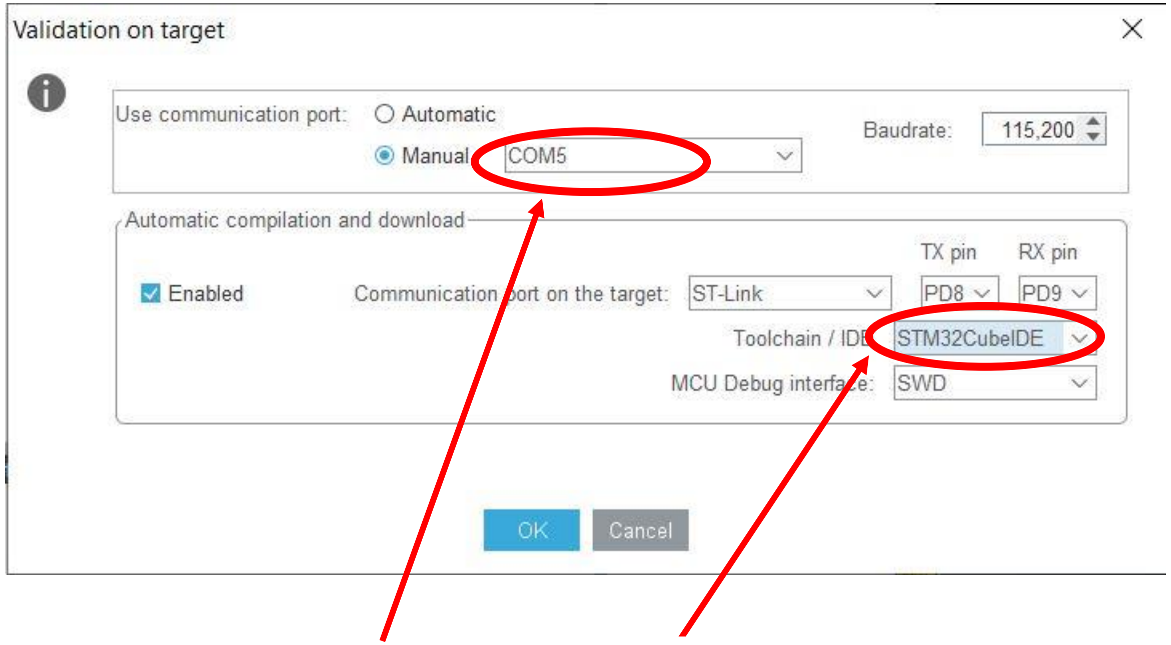
Validation inputs: Random numbers

Validation outputs: None

Complexity: -
Flash occupation: -
RAM: -
Achieved compression: -
Analysis status: -

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

Validate on target: need to have the NUCLEO-H7 plugged.



Use communication port: ○ Automatic ● Manual COM5

Baudrate: 115,200

☑ Enabled

Communication port on the target: ST-Link

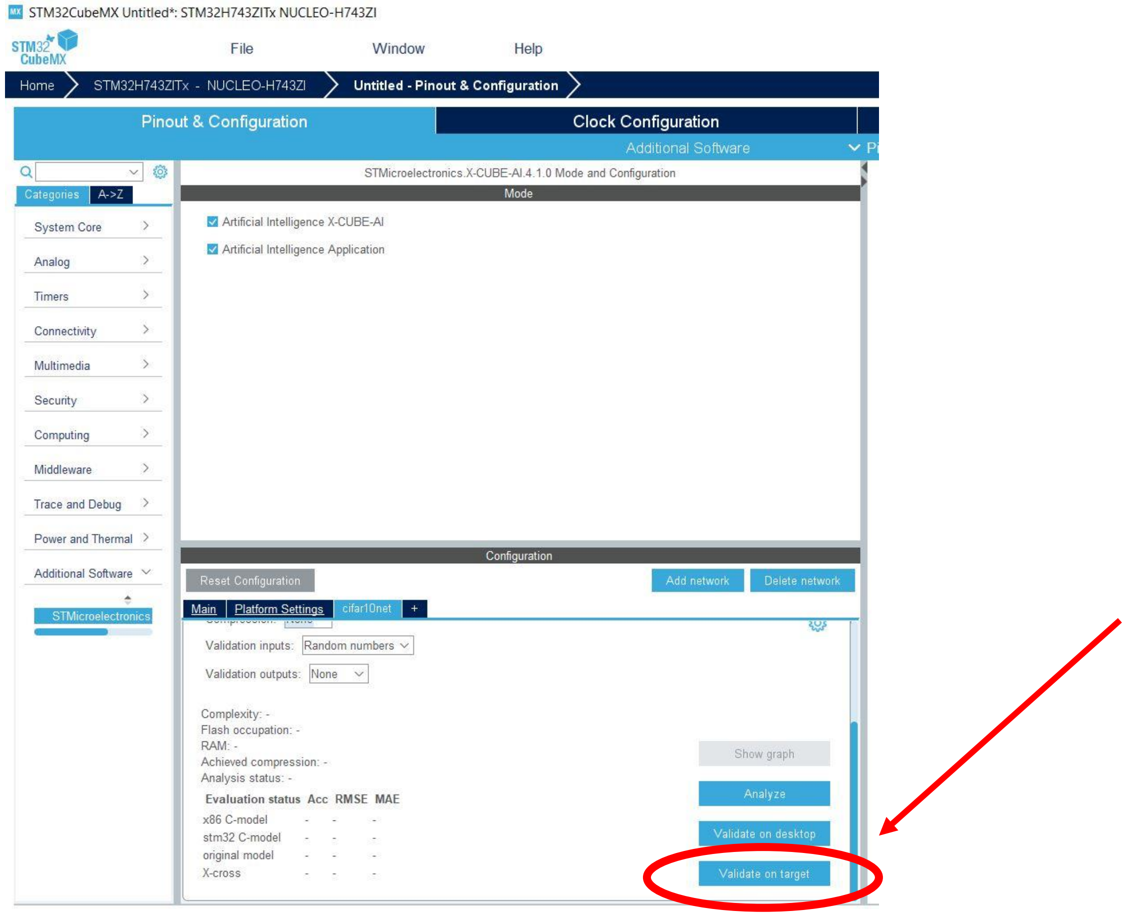
TX pin: PD8 RX pin: PD9

Toolchain / IDE: STM32CubeIDE

MCU Debug interface: SWD

Select correct IDE and COM

STM32CubeMX-AI: actions



☑ Artificial Intelligence X-CUBE-AI

☑ Artificial Intelligence Application

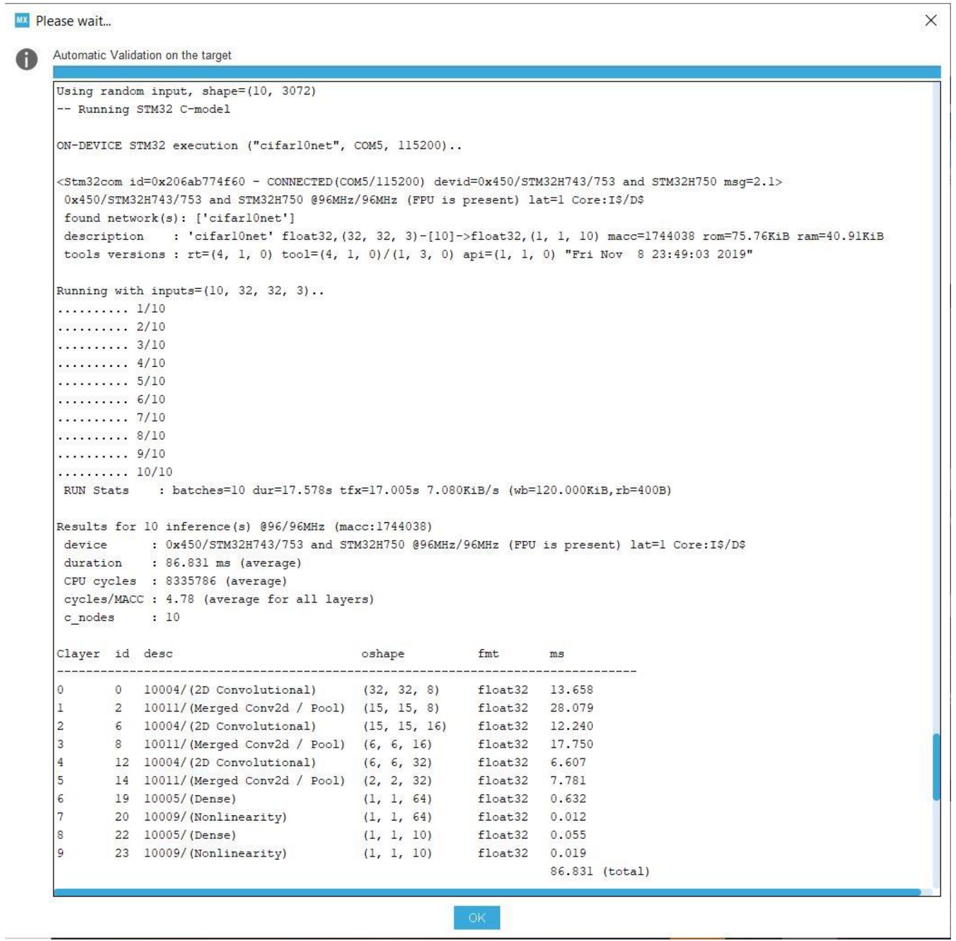
Validation inputs: Random numbers

Validation outputs: None

Complexity: -
Flash occupation: -
RAM: -
Achieved compression: -
Analysis status: -

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

Validate on target: need to have the NUCLEO-H7 plugged.

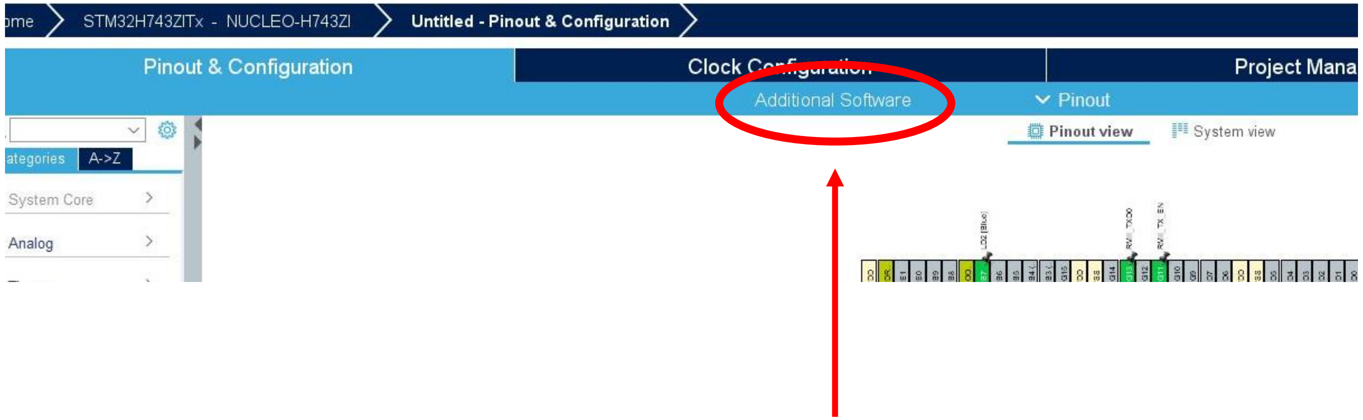


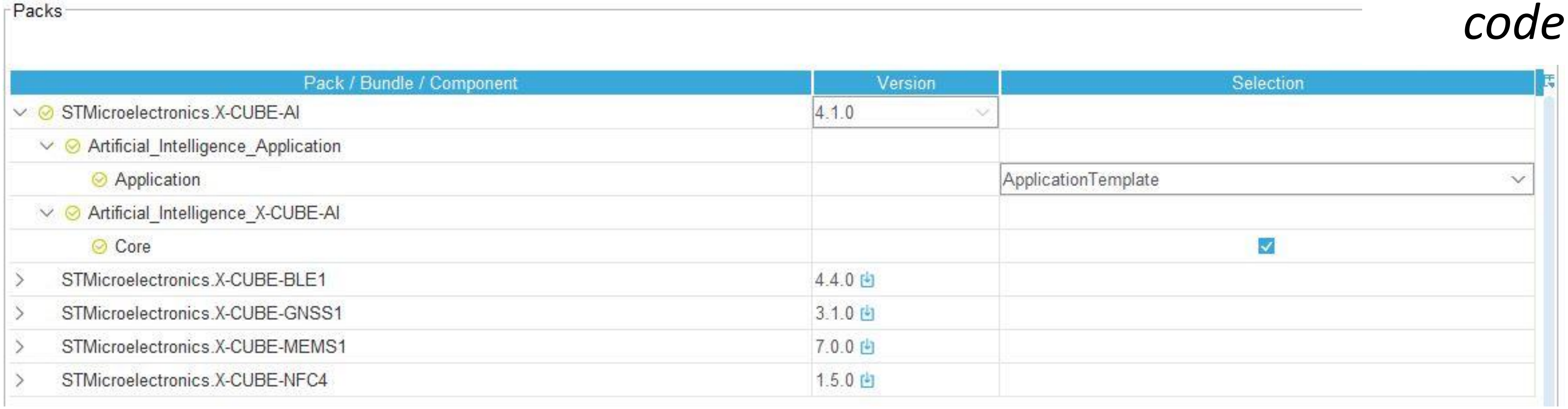
```
Using random input, shape=(10, 3072)
-- Running STM32 C-model

ON-DEVICE STM32 execution ("cifar10net", COM5, 115200)..

<Stm32com id=0x206ab774f60 - CONNECTED(COM5/115200) devid=0x450/STM32H743/753 and STM32H750 msg=2.1>
 0x450/STM32H743/753 and STM32H750 @96MHz/96MHz (FPU is present) lat=1 Core:I$/D$
 found network(s): ['cifar10net']
 description    : 'cifar10net' float32,(32, 32, 3)-[10]->float32,(1, 1, 10) macc=1744038 rom=75.76KiB ram=40.91KiB
 tools versions : rt=(4, 1, 0) tool=(4, 1, 0)/(1, 3, 0) api=(1, 1, 0) "Fri Nov  8 23:49:03 2019"

Running with inputs=(10, 32, 32, 3)..
.......... 1/10
.......... 2/10
.......... 3/10
.......... 4/10
.......... 5/10
.......... 6/10
.......... 7/10
.......... 8/10
.......... 9/10
.......... 10/10
 RUN Stats    : batches=10 dur=17.578s tfx=17.005s 7.080KiB/s (wb=120.000KiB,rb=400B)

Results for 10 inference(s) @96/96MHz (macc:1744038)
 device      : 0x450/STM32H743/753 and STM32H750 @96MHz/96MHz (FPU is present) lat=1 Core:I$/D$
 duration    : 86.831 ms (average)
 CPU cycles  : 8335786 (average)
 cycles/MACC : 4.78 (average for all layers)
 c_nodes     : 10

Clayer  id  desc                          oshape          fmt       ms
--------------------------------------------------------------------------------
0       0   10004/(2D Convolutional)      (32, 32, 8)     float32   13.658
1       2   10011/(Merged Conv2d / Pool)  (15, 15, 8)     float32   28.079
2       6   10004/(2D Convolutional)      (15, 15, 16)    float32   12.240
3       8   10011/(Merged Conv2d / Pool)  (6, 6, 16)      float32   17.750
4       12  10004/(2D Convolutional)      (6, 6, 32)      float32   6.607
5       14  10011/(Merged Conv2d / Pool)  (2, 2, 32)      float32   7.781
6       19  10005/(Dense)                 (1, 1, 64)      float32   0.632
7       20  10009/(Nonlinearity)          (1, 1, 64)      float32   0.012
8       22  10005/(Dense)                 (1, 1, 10)      float32   0.055
9       23  10009/(Nonlinearity)          (1, 1, 10)      float32   0.019
                                                                    86.831 (total)
```

Summary

- Creation of the python network and production of the saved model (.h5)
- Creation of the CubeMX project and validation on NUCLEO-H743ZI
- **Creation of the application for network inference**

STM32CubeMX-AI: application generation



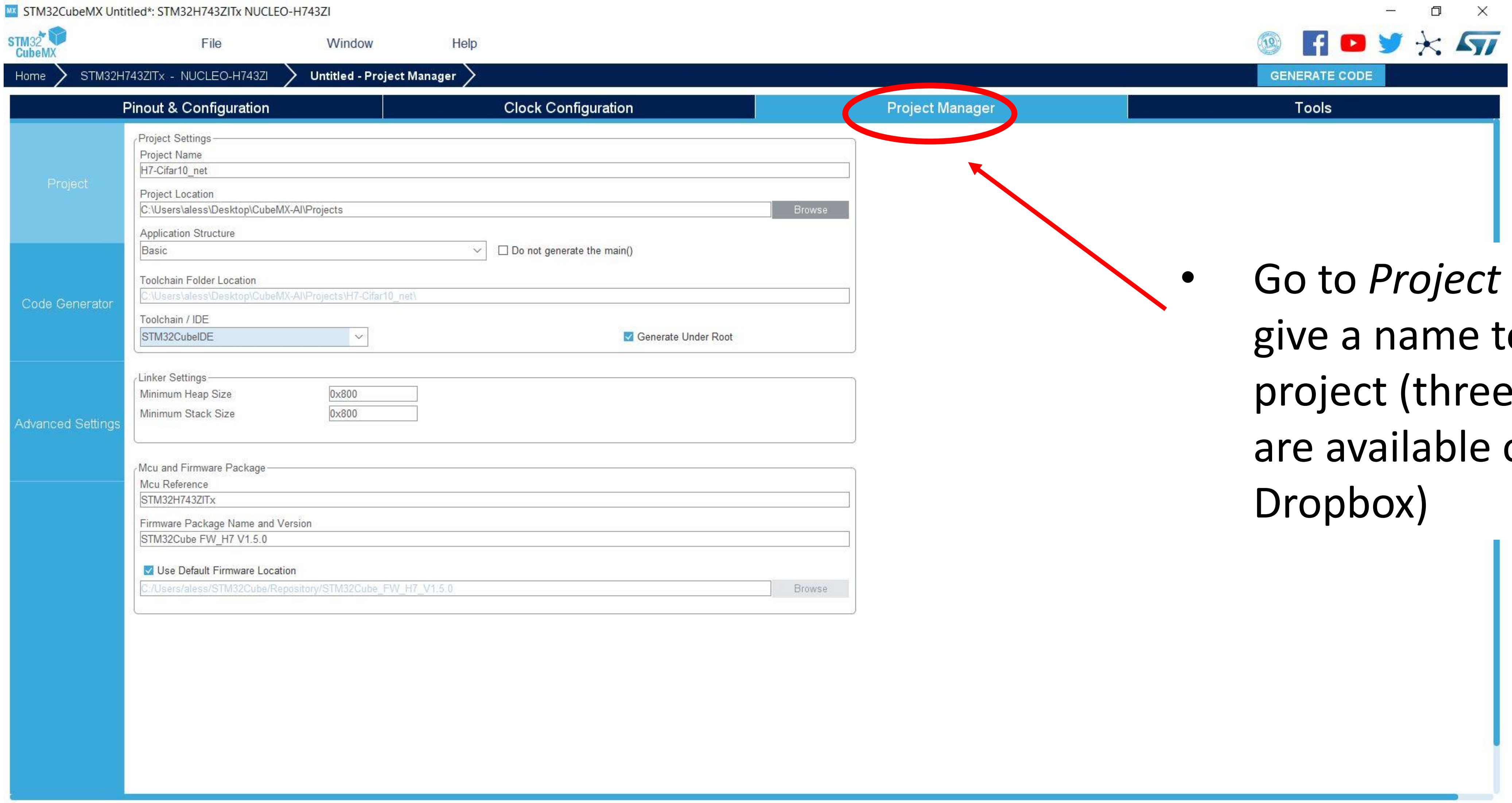


Packs

Pack / Bundle / Component	Version	Selection
STMicroelectronics.X-CUBE-AI	4.1.0	
Artificial_Intelligence_Application		
Application		ApplicationTemplate
Artificial_Intelligence_X-CUBE-AI		
Core		☑
STMicroelectronics.X-CUBE-BLE1	4.4.0	
STMicroelectronics.X-CUBE-GNSS1	3.1.0	
STMicroelectronics.X-CUBE-MEMS1	7.0.0	
STMicroelectronics.X-CUBE-NFC4	1.5.0	

- Change *Validation* to *Application template*. The application template allows to Generate the code using the *Generate code* button.

STM32CubeMX-AI: application generation



Project Settings

Project Name: H7-Cifar10_net

Project Location: C:\Users\aless\Desktop\CubeMX-AI\Projects [Browse]

Application Structure: Basic

☐ Do not generate the main()

Toolchain Folder Location: C:\Users\aless\Desktop\CubeMX-AI\Projects\H7-Cifar10_net\

Toolchain / IDE: STM32CubeIDE

☑ Generate Under Root

Linker Settings

Minimum Heap Size: 0x800

Minimum Stack Size: 0x800

Mcu and Firmware Package

Mcu Reference: STM32H743ZITx

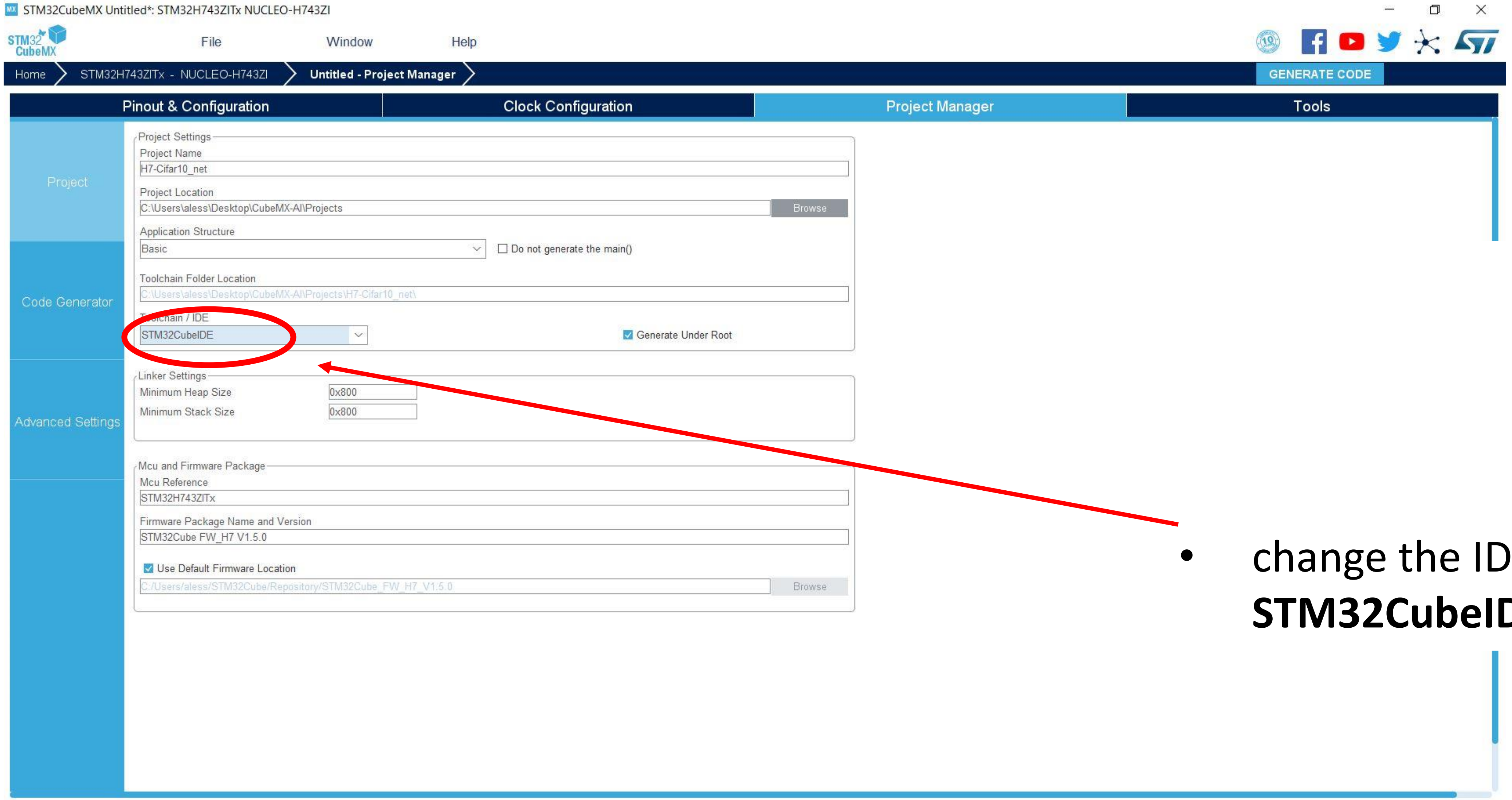
Firmware Package Name and Version: STM32Cube FW_H7 V1.5.0

☑ Use Default Firmware Location

C:/Users/aless/STM32Cube/Repository/STM32Cube_FW_H7_V1.5.0 [Browse]

- Go to *Project Manager*, give a name to the project (three projects are available on Dropbox)

STM32CubeMX-AI: application generation



Project Settings

Project Name: H7-Cifar10_net

Project Location: C:\Users\aless\Desktop\CubeMX-AI\Projects [Browse]

Application Structure: Basic ☐ Do not generate the main()

Toolchain Folder Location: C:\Users\aless\Desktop\CubeMX-AI\Projects\H7-Cifar10_net\

Toolchain / IDE: STM32CubeIDE ☑ Generate Under Root

Linker Settings

Minimum Heap Size: 0x800

Minimum Stack Size: 0x800

Mcu and Firmware Package

Mcu Reference: STM32H743ZITx

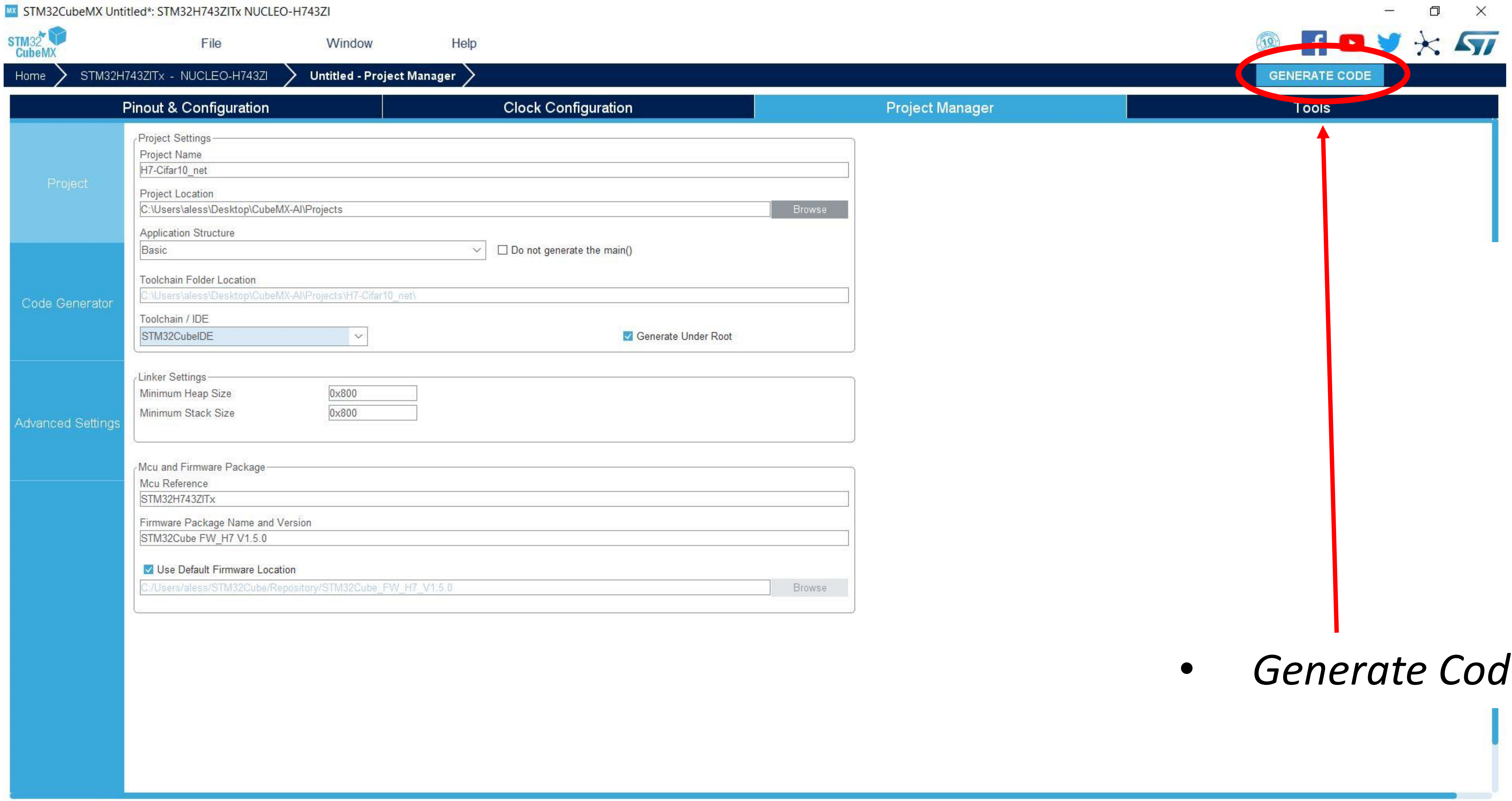
Firmware Package Name and Version: STM32Cube FW_H7 V1.5.0

☑ Use Default Firmware Location

C:/Users/aless/STM32Cube/Repository/STM32Cube_FW_H7_V1.5.0 [Browse]

- change the IDE to **STM32CubeIDE**

STM32CubeMX-AI: application generation



Project Settings

Project Name: H7-Cifar10_net

Project Location: C:\Users\aless\Desktop\CubeMX-AI\Projects

Application Structure: Basic

□ Do not generate the main()

Toolchain Folder Location: C:\Users\aless\Desktop\CubeMX-AI\Projects\H7-Cifar10_net\

Toolchain / IDE: STM32CubeIDE

☑ Generate Under Root

Linker Settings

Minimum Heap Size: 0x800

Minimum Stack Size: 0x800

Mcu and Firmware Package

Mcu Reference: STM32H743ZITx

Firmware Package Name and Version: STM32Cube FW_H7 V1.5.0

☑ Use Default Firmware Location

C:/Users/aless/STM32Cube/Repository/STM32Cube_FW_H7_V1.5.0

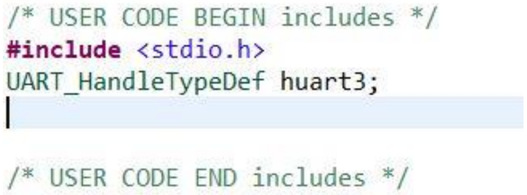
- *Generate Code*

STM32CubeMX-AI Application: add Custom code

- Add Custom code to communicate with the NUCLEO Board and classify with the network
    - USART Receive to receive the image from the PC
    - USART Transmit to send the results back to the PC

1. File to be modified → app_x-cube-ai.c
2. Modifications:
    - Usart declaration (we used USART3 since it is the one linked to the STLINK)
    - Usart initialization
    - Usart communication

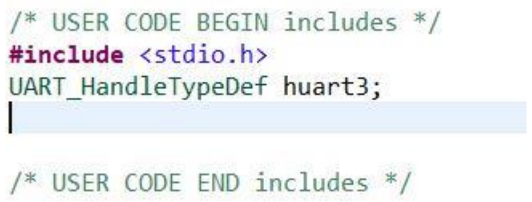
STM32CubeMX-AI Application: add Custom code



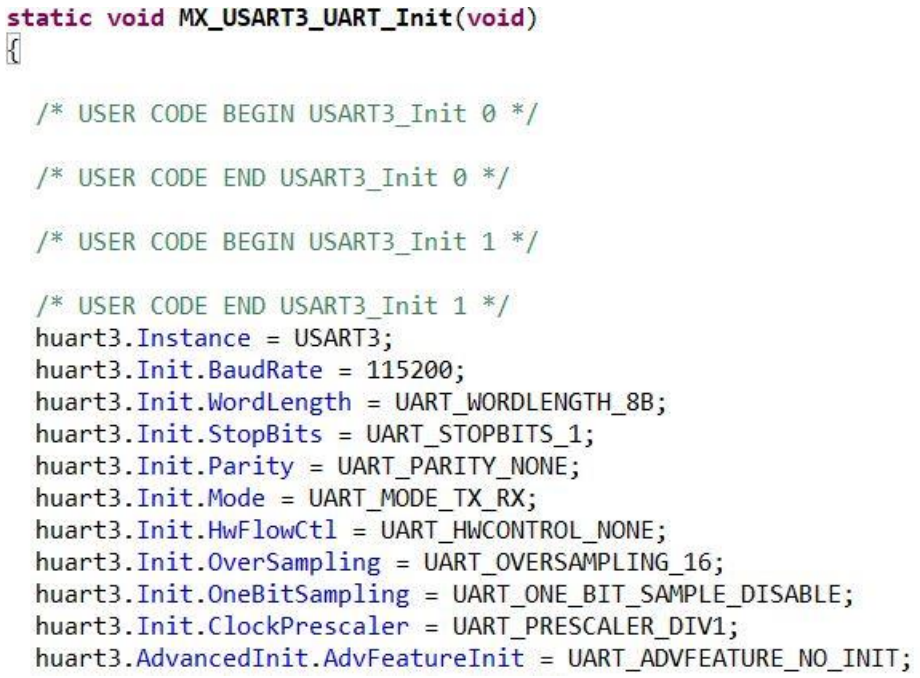
```
/* USER CODE BEGIN includes */
#include <stdio.h>
UART_HandleTypeDef huart3;


/* USER CODE END includes */
```

1. Usart declaration

STM32CubeMX-AI Application: add Custom code



```
/* USER CODE BEGIN includes */
#include <stdio.h>
UART_HandleTypeDef huart3;


/* USER CODE END includes */
```

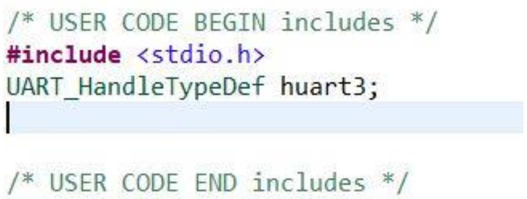


```
static void MX_USART3_UART_Init(void)
{

  /* USER CODE BEGIN USART3_Init 0 */

  /* USER CODE END USART3_Init 0 */

  /* USER CODE BEGIN USART3_Init 1 */

  /* USER CODE END USART3_Init 1 */
  huart3.Instance = USART3;
  huart3.Init.BaudRate = 115200;
  huart3.Init.WordLength = UART_WORDLENGTH_8B;
  huart3.Init.StopBits = UART_STOPBITS_1;
  huart3.Init.Parity = UART_PARITY_NONE;
  huart3.Init.Mode = UART_MODE_TX_RX;
  huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart3.Init.OverSampling = UART_OVERSAMPLING_16;
  huart3.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart3.Init.ClockPrescaler = UART_PRESCALER_DIV1;
  huart3.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
```

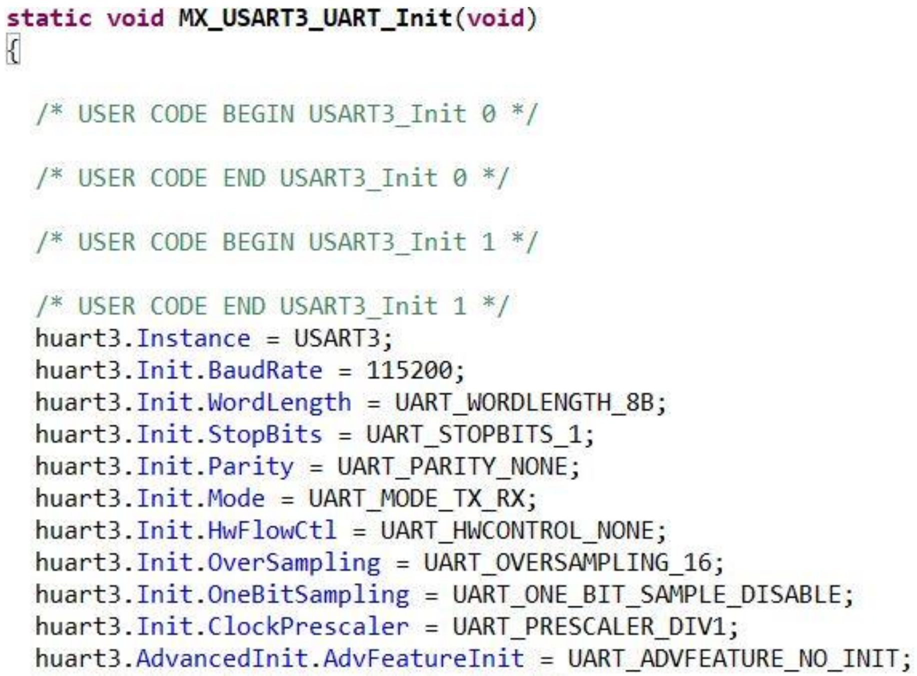
1. Usart declaration
2. Usart init function declaration (copy from *main.c* file)

STM32CubeMX-AI Application: add Custom code

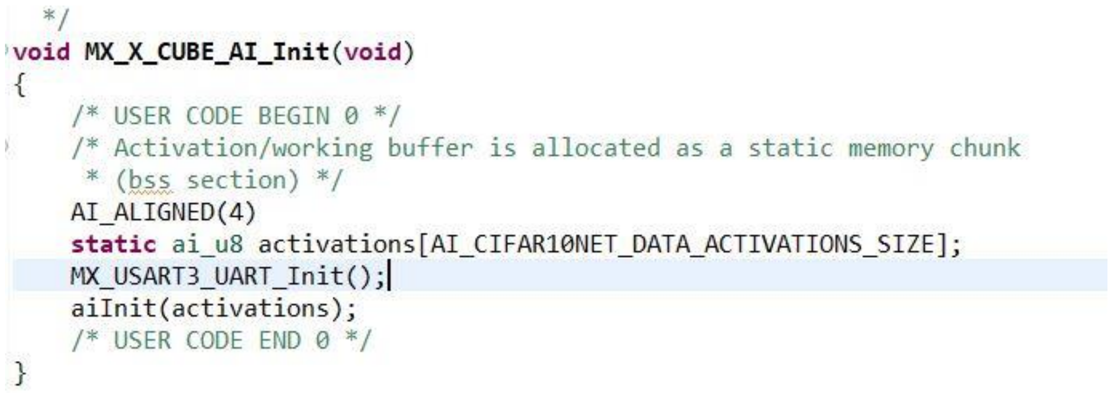
1. Usart declaration
2. Usart init function declaration (copy from *main.c* file)
3. Usart initialization in MX_X_CUBE_AI_Init



```
/* USER CODE BEGIN includes */
#include <stdio.h>
UART_HandleTypeDef huart3;


/* USER CODE END includes */
```



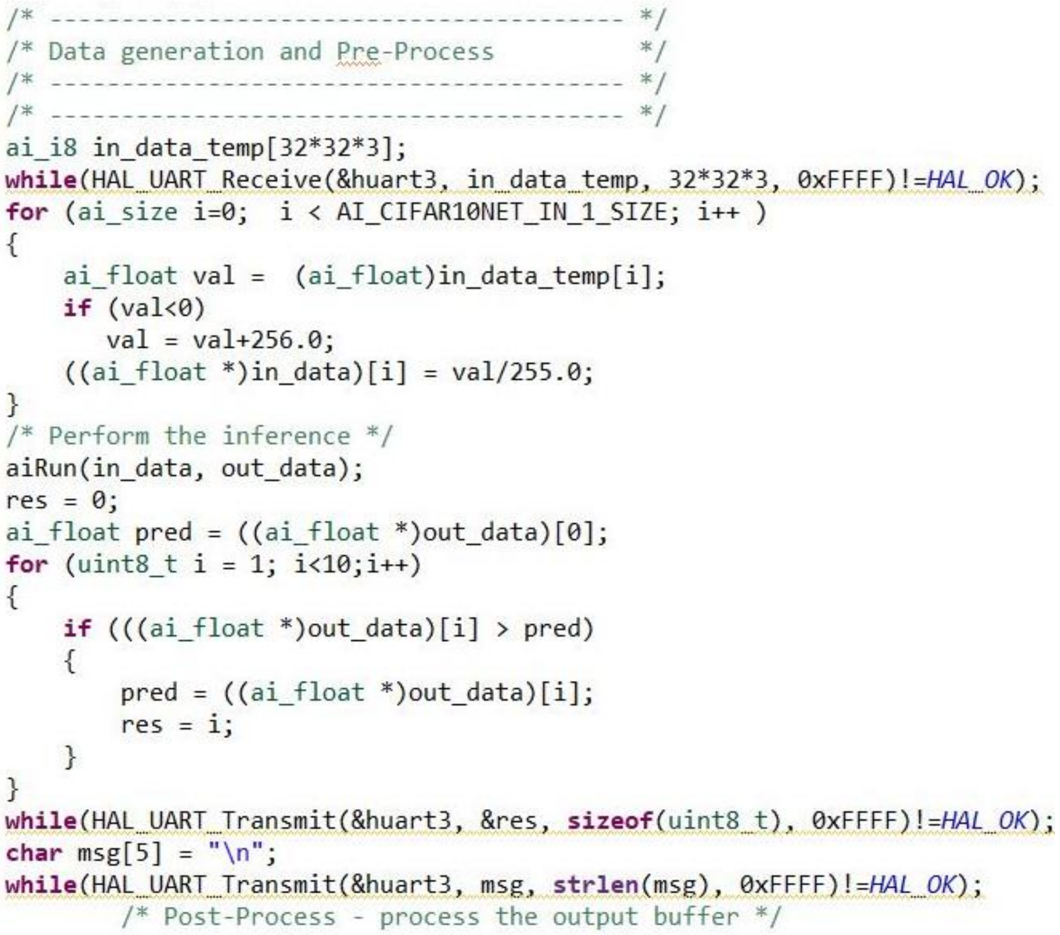
```
static void MX_USART3_UART_Init(void)
{

  /* USER CODE BEGIN USART3_Init 0 */

  /* USER CODE END USART3_Init 0 */

  /* USER CODE BEGIN USART3_Init 1 */

  /* USER CODE END USART3_Init 1 */
  huart3.Instance = USART3;
  huart3.Init.BaudRate = 115200;
  huart3.Init.WordLength = UART_WORDLENGTH_8B;
  huart3.Init.StopBits = UART_STOPBITS_1;
  huart3.Init.Parity = UART_PARITY_NONE;
  huart3.Init.Mode = UART_MODE_TX_RX;
  huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart3.Init.OverSampling = UART_OVERSAMPLING_16;
  huart3.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart3.Init.ClockPrescaler = UART_PRESCALER_DIV1;
  huart3.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
```



```
  */
void MX_X_CUBE_AI_Init(void)
{
    /* USER CODE BEGIN 0 */
    /* Activation/working buffer is allocated as a static memory chunk
     * (bss section) */
    AI_ALIGNED(4)
    static ai_u8 activations[AI_CIFAR10NET_DATA_ACTIVATIONS_SIZE];
    MX_USART3_UART_Init();
    aiInit(activations);
    /* USER CODE END 0 */
}
```

STM32CubeMX-AI Application: add Custom code

1. Usart declaration
2. Usart init function declaration (copy from *main.c* file)
3. Usart initialization in MX_X_CUBE_AI_Init
4. MX_X_CUBE_AI_Process → change the input data stream from random data to USART transmitted data



```c
/* --------------------------------------- */
/* Data generation and Pre-Process         */
/* --------------------------------------- */
/* --------------------------------------- */
ai_i8 in_data_temp[32*32*3];
while(HAL_UART_Receive(&huart3, in_data_temp, 32*32*3, 0xFFFF)!=HAL_OK);
for (ai_size i=0;  i < AI_CIFAR10NET_IN_1_SIZE; i++ )
{
    ai_float val =  (ai_float)in_data_temp[i];
    if (val<0)
       val = val+256.0;
    ((ai_float *)in_data)[i] = val/255.0;
}
/* Perform the inference */
aiRun(in_data, out_data);
res = 0;
ai_float pred = ((ai_float *)out_data)[0];
for (uint8_t i = 1; i<10;i++)
{
    if (((ai_float *)out_data)[i] > pred)
    {
        pred = ((ai_float *)out_data)[i];
        res = i;
    }
}
while(HAL_UART_Transmit(&huart3, &res, sizeof(uint8_t), 0xFFFF)!=HAL_OK);
char msg[5] = "\n";
while(HAL_UART_Transmit(&huart3, msg, strlen(msg), 0xFFFF)!=HAL_OK);
        /* Post-Process - process the output buffer */
```

STM32CubeMX-AI Application: load the code

- Load the code using the STM32CubeIDE on the board, which will be ready to run the network, waiting for an input image.