DOI: 10.35598/mcfpga.2023.020

Neuron Networks Design in STM32 Cube

Oleksandr Vorgul
ORCID 0000-0002-7659-8796
dept. Microprocessor Technologies and Systems
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
oleksandr.vorgul@nure.ua

Iryna Svyd
ORCID 0000-0002-4635-6542
dept. Microprocessor Technologies and Systems
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
iryna.svyd@nure.ua

***Abstract*—This article is devoted to analyzing the possibilities of using a neural network in a project based on STM32 microcontrollers. Support for many different libraries is discussed. The limited RAM size can be mitigated by using external flash memory connected to the USB port.**

Keywords—AI, neural network, STM32.

I. Introduction

Today, the use of neural networks in technical applications is no longer surprising. In various projects, these tasks can be implemented using a variety of methods, including with the support of a neural network [1-5]. In some cases, it gives an advantage in the efficiency of solving the problem [1].

In terms of general applications for artificial intelligence, it seems that the best field for AI is to intelligently search very large data sets and format information, for example, from text form to visual form; or processing of sound, speech, etc. On the contrary, st.com in its ever-expanding portfolio already has several utilities for using pre-trained neural networks to implement user projects.

II. State of the Art in the Neuron Networks

To create a working neural network, one should complete a list of steps [1, 6]. The main stages of creating an artificial neural network:

- Data collection.
- Preprocessing.
- Building and training the model.
- Quality analysis and interpretation of the model.

Even in the conditions of the modern proliferation of computers, the implementation of the listed stages is associated with a significant expenditure of time and money, primarily human ones.

One cannot get rid of the data collecting stage. It can be automated, but it is strongly recommended that the data should be preprocessed and labeled for training purposes. It is this stage that takes time and supervising. These two stages are considered as a data preparation [7].

As for the building and training the network, this process is not completely formalized, but strong support can be easily found. There are well-built programs of deep learning. Among them there are propriety software and free software. It is Keras, Microsoft Cognitive Toolkit, ML.NET, OpenNN, PyTorch, TensorFlow, ONNX to name some from free software (Fig. 1).

For example, using link of Kears and Tensor Flow one can prepare well-structured and trained and neural network model, but it is not enough to start a technical project.

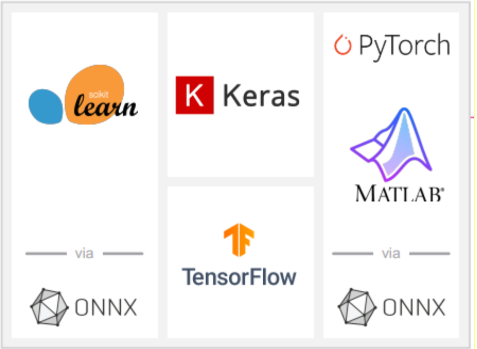


Fig. 1. AI model libraries that can be supported by current ST utilites.

III. ST Offers

To start your work with hardware, one should have the utility. It is better to be some good one like STM32 Cube AI. And to load your pretrained model, it is better should be optimized and validated.

So, the STM32Cube AI. It can be used together with CubeMX or through command line interface. It provides the tool to build step by step a complete Artificial Intelligence (AI) IDE-based project for STM32 microcontrollers with automatic conversion of pretrained Neural Networks (NN) and integration of the generated optimized library. The X-CUBE-AI Expansion Package is fully integrated with the STM32CubeMX tool. Its user manual also describes optional add-on AI test applications or utilities for AI system performance and validation.

It is announced that it is supported by various boards of ST production, not only STM32G4, STM32L4, STM32L4+, STM32L5, STM32F7, STM32H7 lineage, but by STM32WB, or STM32WL and even STM32F0, STM32F3, STM32F4and STM32G0.

Its core engine provides an automatic and advanced NN mapping tool to generate and deploy an optimized and robust C-model implementation of a pretrained Neural Network (DL model) for the embedded systems with limited and constrained hardware resources. The generated STM32 NN library (both specialized and generic parts) can be directly integrated in an IDE project or makefile-based build system. A well-defined and specific inference client API is also exported to develop a client AI-based application. Various frameworks (DL toolbox) and layers for Deep Learning are supported. All X-CUBE-AI core features are available through a complete and unified Command Line Interface (console level) to perform the main steps to analyze, validate, and generate an optimized NN C-library for STM32. It provides also a post-training quantization support for the Keras model [7].

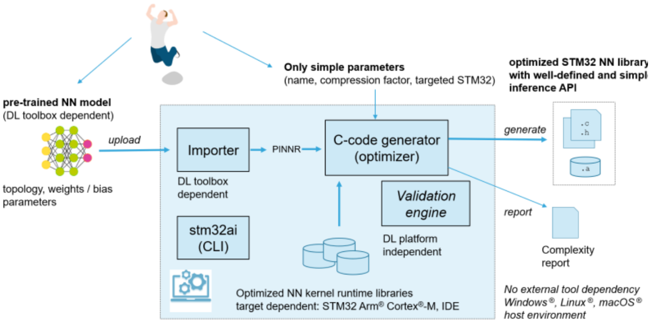

pre-trained NN model
(DL toolbox dependent)
upload
topology, weights / bias parameters
Only simple parameters
(name, compression factor, targeted STM32)
optimized STM32 NN library with well-defined and simpl inference API
Importer
DL toolbox dependent
PINNR
C-code generator (optimizer)
generate
.c
.h
.a
Validation engine
DL platform independent
report
Complexity report
stm32ai (CLI)
Optimized NN kernel runtime libraries
target dependent: STM32 Arm® Cortex®-M, IDE
No external tool dependency
Windows®, Linux®, macOS®
host environment

Fig. 2. X-CUBE-AI core engine.

To keep the whole system on diet, X-CUBE-AI code generator can be used to generate and deploy a prequantized 8-bit fixed-point/integer Keras model and the quantized TensorFlow™ Lite model. For the Keras model, a reshaped model file (h5*) and a proprietary tensor-format configuration file (json) are required [7].

The code generator quantizes weights and bias, and associated activations from floating point to 8-bit precision. These are mapped on the optimized and specialized C implementation for the supported kernels. Otherwise, the floating-point version of the operator is used and float-to-8-bit and 8-bit-to-float convert operators are automatically inserted. The objective of this technique is to reduce the model size while also improving the CPU and hardware accelerator latency (including power consumption aspects) with little degradation in model accuracy [7].

The component X-Cube-AI got variety of controls (Fig. 3)

From the user point of view, the integration of the X-CUBE-AI Expansion Package can be considered as the addition of a peripheral or middleware software component. On top of X-CUBE-AI core, the following main functionalities are provided:

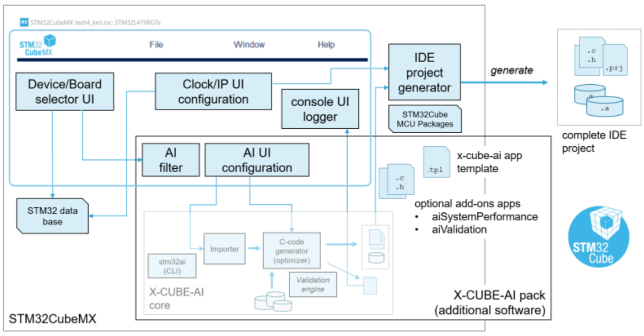
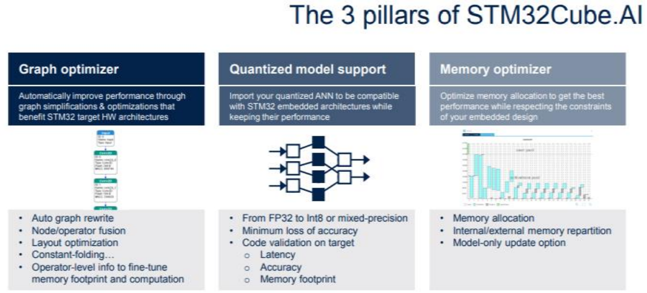


Fig. 3. X-CUBE-AI core in STM32CubeMX.

- MCU filter selector is extended with an optional specific AI filter to remove the devices that do not have enough memory. If enabled, STM32 devices without Arm® Cortex®-M4, -M7, or -M33 core are directly filtered out.
- Provides a complete AI UI configuration wizard allowing the upload of multiple DL models. Includes a validation process of the generated C code on the desktop PC and on the target.
- Extends the IDE project generator to assist the generation of the optimized STM32 NN library and its integration for the selected STM32 Arm® Cortex®-M core and IDE.
- Optional add-on applications allow the generation of a complete and ready-to-use AI test application project including the generated NN libraries. The user must just have imported it inside the favorite IDE to generate the firmware image and program it. No additional code or modification is requested from the end user.
- Generation using STM32Cube.AI runtime or TensorFlow™ Lite for Microcontrollers runtime when the Neural Network file is a TensorFlow™ Lite file [7].

The plan is that optimized C-code may be obtained and loaded into some ST microcontroller. By this way one can achieve neural network realization on ST micro controller.

But it is not an end of the story. Once one has a code one has a chance to improve it.


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 or mixed-precision
- Minimum loss of accuracy
- Code validation on target
  - Latency
  - Accuracy
  - Memory footprint

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

Fig. 4. More advantages.

Usually, the ST utilities are well documented and X-Cube AI is not an exception. There is step by step documentation [7], that starts from installing the X-Cube – AI, considering an idea of neural network, taking the data from the open source and implementing it on F410 board (not H747). Among actions, in a process of completing the project, the AI part announced that there is not enough memory to store even optimized NN parameters into. And in order to fix the issue one can attach USB flesh memory module. Quite a situation.

More, there was an interesting workshop about X-Cube-AI utilization. Again, it covers the NN building from scratch to its installing and optimization and further check on practice [8, 9].

Among projects, one project is Fast Downsampling MobileNet Food Recogninion on STM32H747 Discovery board. Its facilities were:

A. Neural Network

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

B. Implementation

- Exploits Camera in continuous mode 5.5 fps or one shot.
- 18 food classes.
- Pre-processing: rescaling from 640x480 to 224x224 RGB 8 bit image.

C. The results obtained are: STM32 Cube.AI NN

- Memory footprint: 205 KB RAM, 191 KB Flash.

D. 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).

This accuracy seems to be not flawless but quite useful for purposes of recognition.

Also, the workshop announced FP-AI sensing support through practical example (Fig. 5) [10].

FP-AI-SENSING1 is an STM32Cube function pack featuring examples that let you connect your IoT node to a smartphone via BLE and use a suitable Android™ or iOS™ application, like the STBLESensor app, to configure the device. The package enables advanced applications such as human activity recognition or audio scene classification, on the basis of outputs generated by neural networks (NN). The NN are implemented by a multi-network library supporting both floating and fixed point arithmetics, generated by the X-CUBE-AI extension for STM32CubeMX tool. The NN provided in this package are just examples of what can be achieved by combining the output of X-CUBE-AI with connectivity and sensing components from ST.

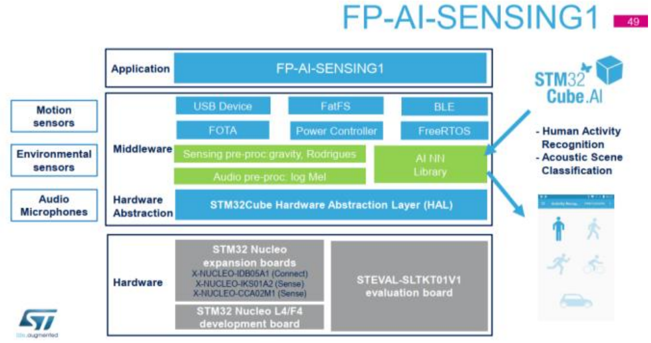


Fig. 5. FP-AI Sensing.

The package comes with an AI utility for data logging and annotation on SD card. You can record the data from the sensors and define which classes or events to record. With the recorded annotated data, you can train your own neural network on your PC/GPU/cloud, get the model, use X-CUBE-AI extension for STM32CubeMX tool for conversion, and then run it on the STM32 platform.

This package, together with the suggested combination of STM32 and ST devices, can be used to develop specific wearable AI applications, industrial predictive maintenance applications, smart things and building applications in general, where ultra-low power consumption is a key requirement.

The software runs on the STM32 microcontroller and includes all the necessary drivers for the STM32 Nucleo development board and expansion boards, as well as for the STEVAL-STLKT01V1 and STEVAL-MKSBOX1V1 evaluation boards and the B-L475E-IOT01A STM32L4 Discovery kit IoT node.

The paper materials span all the stages of NN model creation: collection of data, its preprocessing, model creation in Keras, selecting parameters of the model, data preparation for training, model creation in Python, policies of splitting the dataset onto training validation and exam parts, analyzing the model obtained. It is during evaluation of the model. Some optimization of the model can be involved.

Its Features:

- Complete firmware to develop an IoT node with BLE connectivity, digital microphone, environmental and motion sensors, and perform real-time monitoring of sensors and audio data.
- Middleware library generated thanks to STM32CubeMX extension called X-CUBE-AI, featuring example implementation of neural networks for realtime human activity recognition (HAR) and acoustic scene classification (ASC) applications.
- Multi-network support: concurrent execution of several neural networks.
- AI utility for data logging and annotation on SD card or QSPI Flash memory.
- Ultra-low power implementation based on the use of an RTOS.

- Compatible with STBLESensor application for Android/iOS, to perform sensor data reading, audio and motion algorithm feature demo, and firmware update over the air (full and partial FOTA).
- Easy portability across different MCU families, thanks to STM32Cube.
- Free, user-friendly license terms.

IV. Conclusion

So, it is practically possible to implement a neural network into a project on a not very powerful microprocessor, which has built-in support for using an already trained pre-trained neural network. Optimization of the parameters of such a network, implemented in the component, is very important.

Impress by the support for multiple "neural blocs", the adjustment of multiple parameters for training, the evaluation of the model and its optimization after training, both in a general sense and to improve compatibility with the microprocessor system.

It is not entirely clear whether additional training of the neural network is implemented during operation in the microprocessor system, or maybe this is too much?

And there is one more tool of ST. Automated ML software for end-to-end Edge AI solution design on STM32. But it will be to need another research.

References

[1] Neural Networks and Deep Learning – http://neuralnetworksanddeeplearning.com/chap5.html

[2] Луценко О. В. Використання FPGA для реалізації штучної нейронної мережі / О. В. Луценко, В. С. Чумак // Автоматизація, електроніка та робототехніка. Стратегії розвитку та інноваційні технології : матеріали IV форуму, 24–25 листопада 2022 р. – Харків : ХНУРЕ, 2022. – С. 26-27.

[3] Чумак В. С. Реализация структуры нейронных сетей на FPGA / Чумак В.С., Свид І.В. // Наука, технології, інновації: тенденції розвитку в Україні та світі: матеріали міжнародної студентської наукової конференції, 17 квітня, 2020 рік. – Харків, Україна: Молодіжна наукова ліга. –Т.2– С. 30-32.

[4] Iryna Svyd, Oleksandr Vorgul, Valerii Semenets, Oleg Zubkov, Valeriia Chumak, Natalia Boiko. Special Features of the Educational Component “Design of Devices on Microcontrollers and FPGA”. // II International Scientific and Practical Conference Theoretical and Applied Aspects of Device Development on Microcontrollers and FPGAs (MC&FPGA), Kharkiv, Ukraine, 2020, pp. 55-57. doi: 10.35598/mcfpga.2020.017.

[5] Чумак В. С. Применение FPGA при реализации искусственной нейронной сети для информационных систем. Науковий керівник: Свид І. В. // Авіація, промисловість, суспільство : матеріали II Міжнар. наук.-практ. конф., (м. Кременчук, 12 трав. 2021 р.) : у 2 ч. / МВС України, Харків. нац. ун-т внутр. справ, Кременчуц. льотний коледж. – Харків : ХНУВС, 2021. – Ч. 1. – С. 109-111.

[6] Безрук В.М., Свид І.В., Корсун І.В. Нейронні технології в телекомунікаціях та системах управління: навч. посібник с грифом МОН. Харків, СМІТ, 2008. 230 с.

[7] UM2526 Getting started with X-CUBE-AI Expansion Package for Artificial Intelligence (AI).

[8] Implementing Neural Networks on STM32 https://www.st.com/content/st_com/en/support/learning/stm32-education/stm32-moocs/STM32Cube AI_workshop_MOOC.html.

[9] Програмування мікроконтролерів STM32 в середовищі STM32CubeIDE в прикладах і задачах: Навч. посіб. / О. В. Зубков, І. В. Свид, О. В. Воргуль, В. В. Семенець. Дніпро : ЛІРА ЛТД, 2022. 144 с.

[10] https://www.st.com/en/embedded-software/fp-ai-sensing1.html#:~:text=FP%2DAI%2DSENSING1%20is%20an,app%2C%20to%20configure%20the%20device.