Topic proposal (Draft)

JUAN CARLOS REYES ANDRADE, ICS

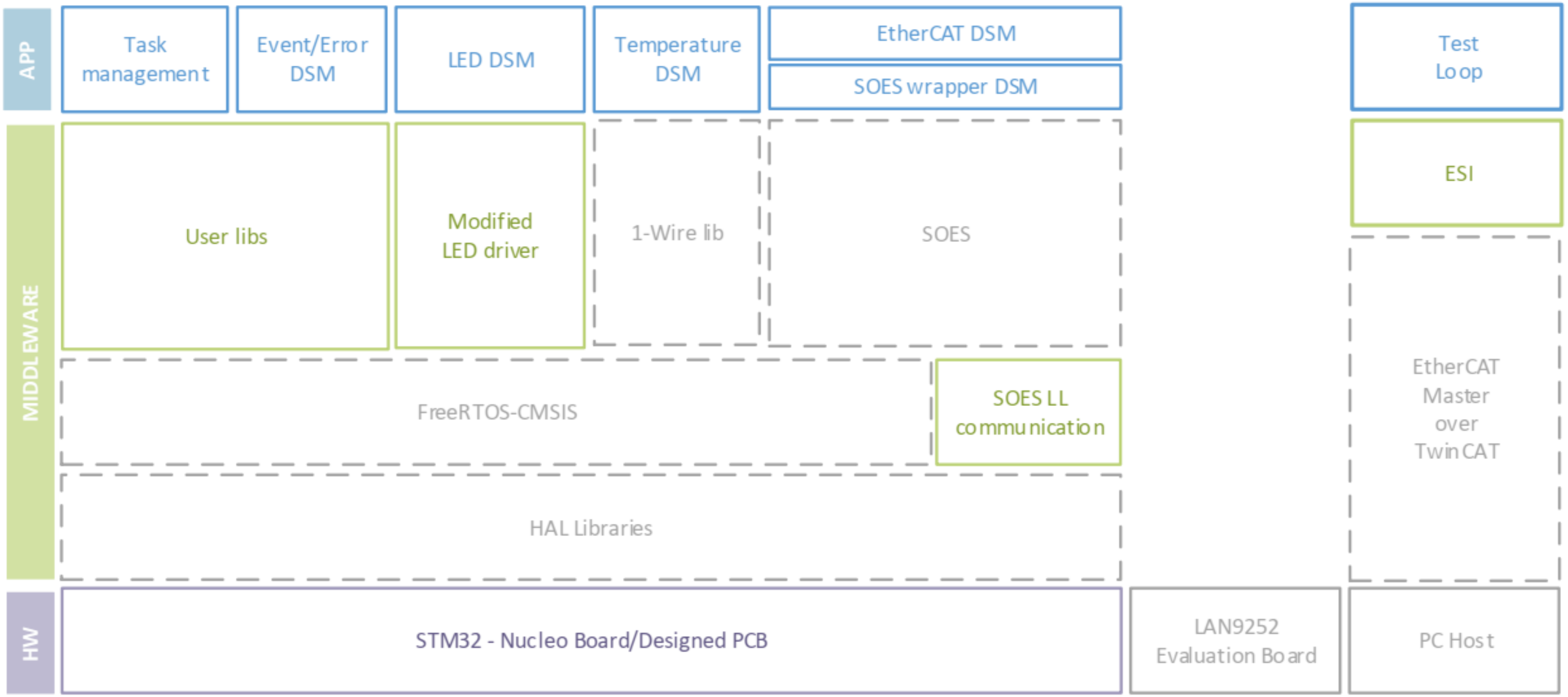
2021.01.18

Axis communication hub

Current structure, highlighting network related tasks, event handler and peripheral control-related tasks.

State machines that are verifiable through software, e.g., UPPAL. Verification software for FSMs

Axis Communication Hub



APP

Task management

Event/Error DSM

LED DSM

Temperature DSM

EtherCAT DSM

SOES wrapper DSM

Test Loop

MIDDLEWARE

User libs

Modified LED driver

1-Wire lib

SOES

ESI

FreeRTOS-CMSIS

SOES LL communication

EtherCAT Master over TwinCAT

HAL Libraries

HW

STM32 - Nucleo Board/Designed PCB

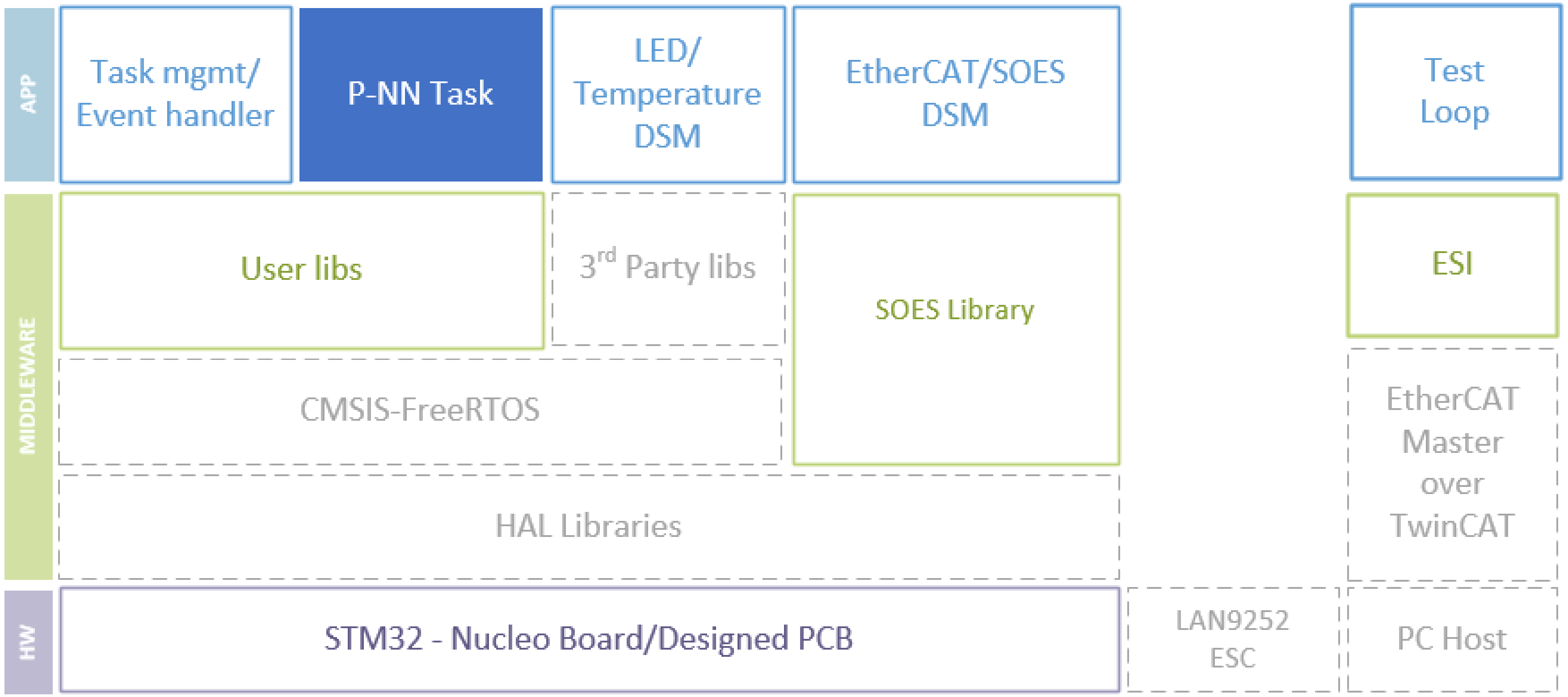
LAN9252 Evaluation Board

PC Host

Structure with new feature

Proposed structure with highlighted Neuronal Network running as a task. Other tasks merged only for space reasons.

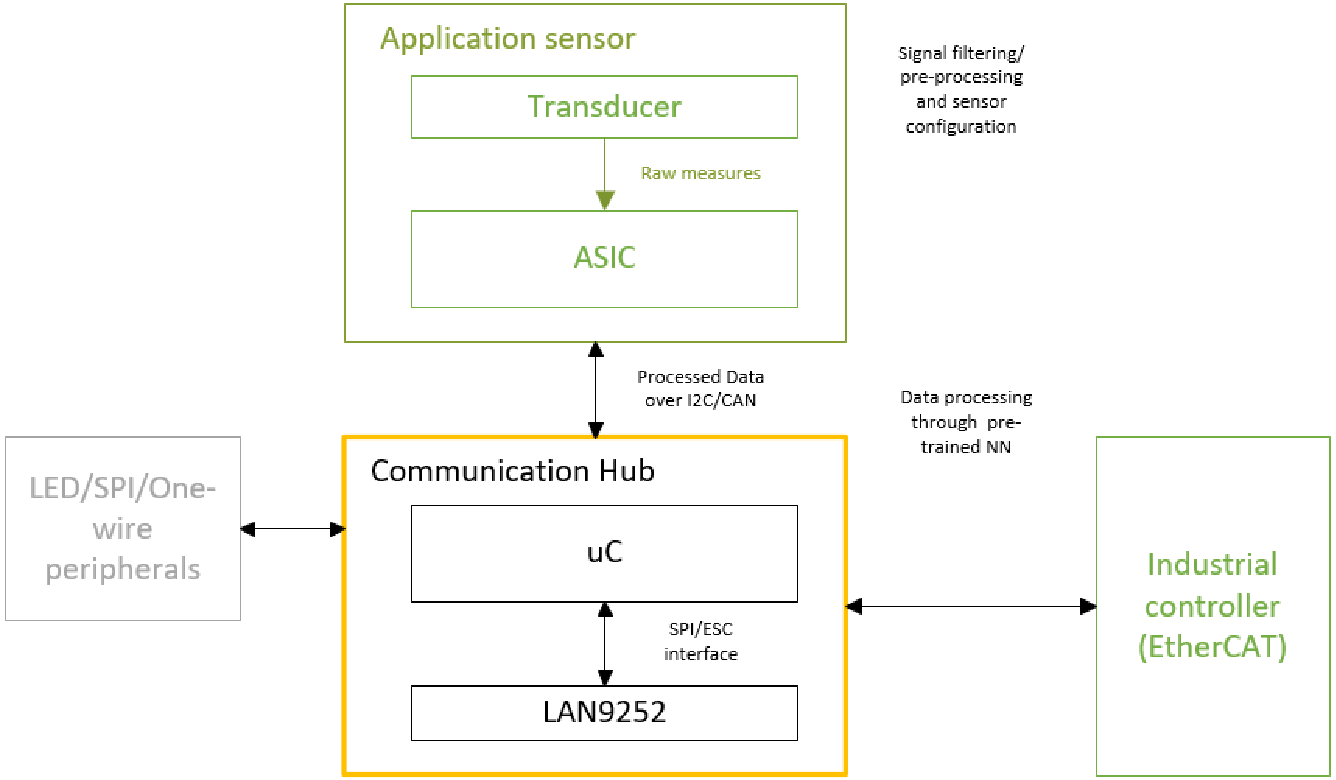
Highlighted block can be adapted to any functionality. Nonetheless, a NN is commonly created out of higher-level tools.



Data flow structure

Proposal assumes a given and high reliability of the sensor and industrial controller.

Whereas the communication hub’s reliability depends on the right execution of its internal features to not affect the “reliability chain”.

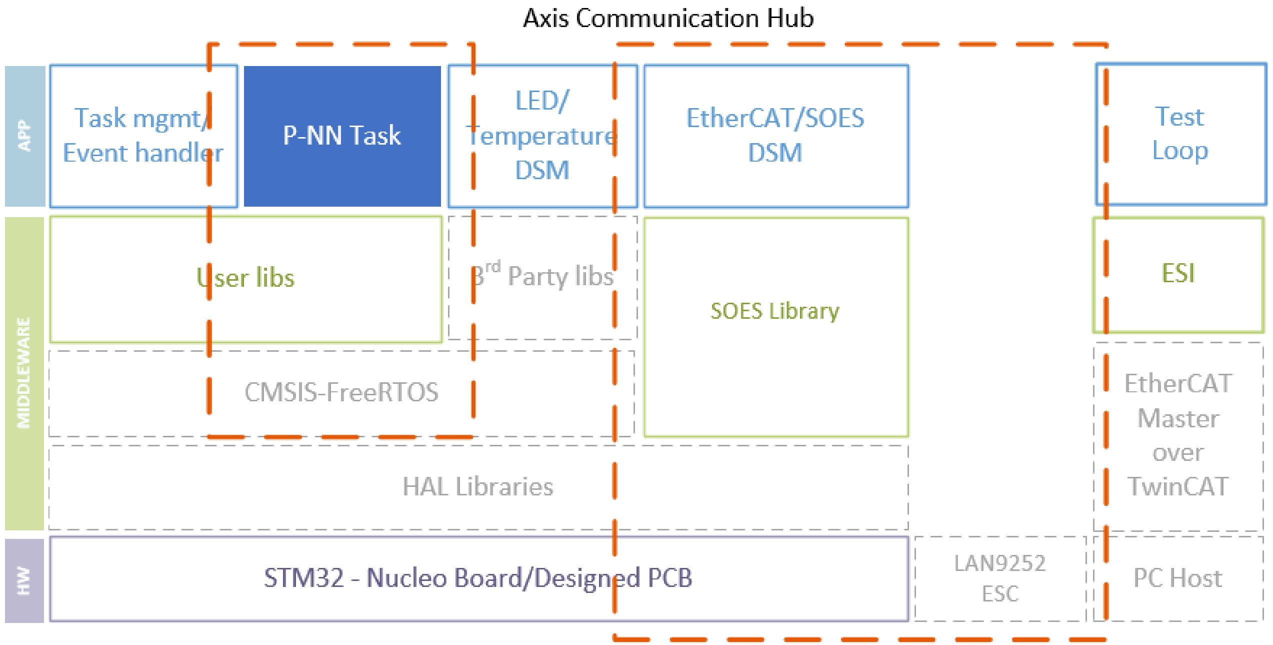

Application sensor
Transducer
Raw measures
ASIC
Signal filtering/ pre-processing and sensor configuration
Processed Data over I2C/CAN
Data processing through pre-trained NN
LED/SPI/One-wire peripherals
Communication Hub
uC
SPI/ESC interface
LAN9252
Industrial controller (EtherCAT)

V&V techniques

Verification, validation and testing techniques could be really demanding according to the formality level.

The firmware implementation can be analyzed in parts.

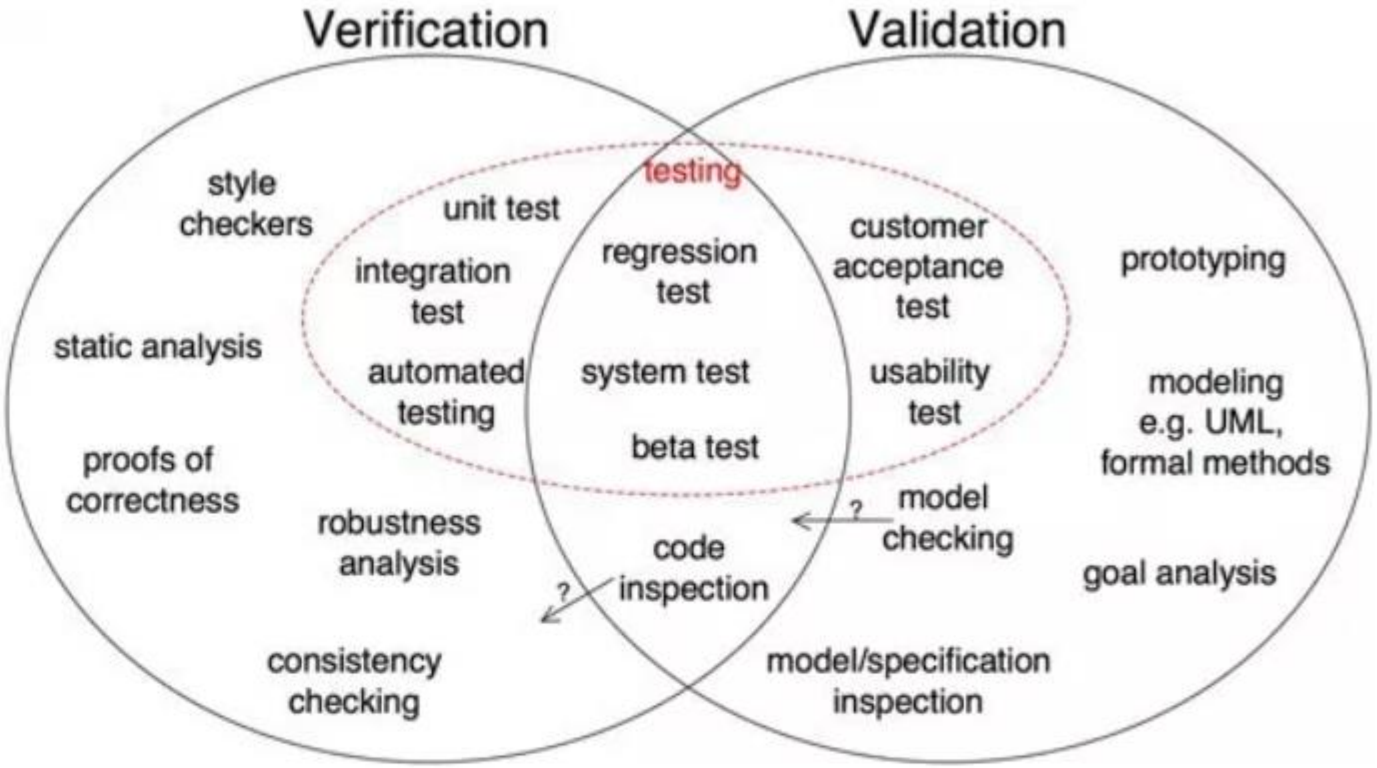
Proposed the Network-related tasks and the NN-related tasks.



V&V

Validation is concerned with checking that the system will meet the customer's actual needs, while verification is concerned with whether the system is well-engineered, error-free, and so on.

Set of tools according to the nature of what is going to be validated or verify and the level of formality.



Proposals

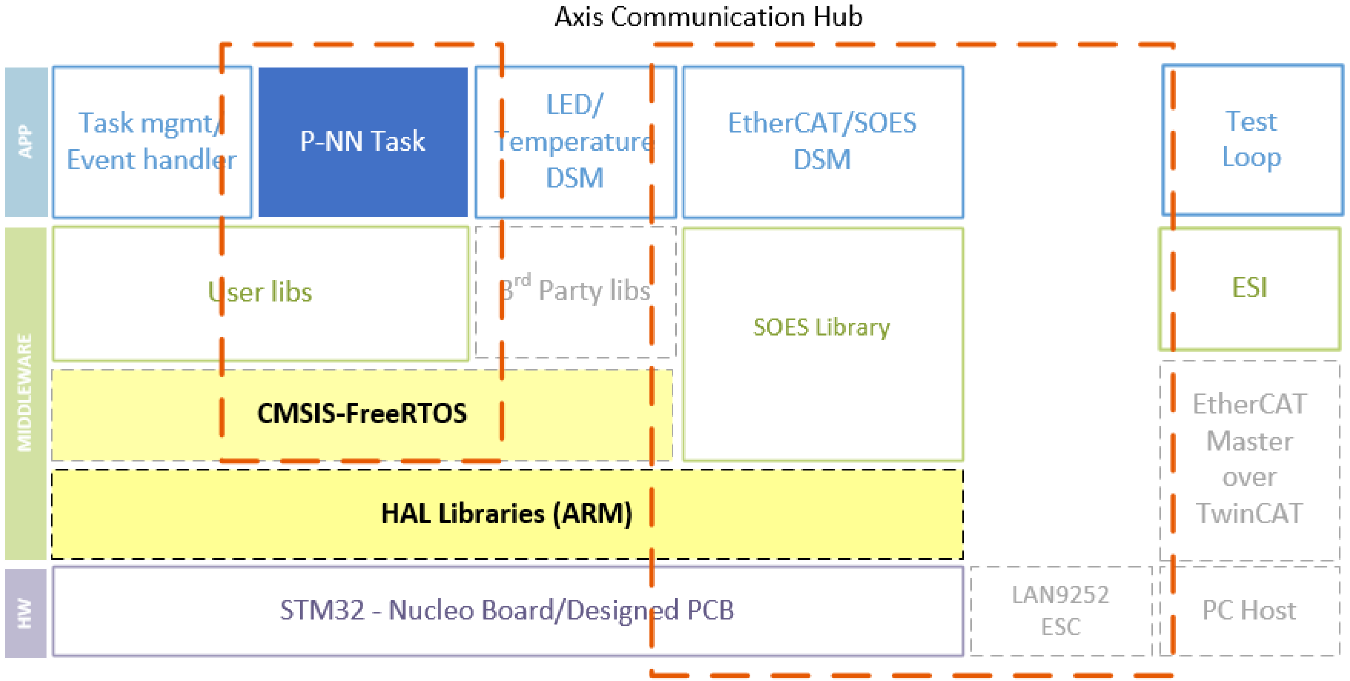
Rely on 3 topics

Software for embedded systems, software verification and machine learning.

Levels of formality within proposals:

1. Formal verification of NNs (only the NN task)
2. Verification of FSMs in embedded systems (Network tasks + P-NN as “black box”)
3. Documentation of dependability analysis according to standards (hardware and less software oriented)

Flexibility



Further steps:

1. **Go through documentation:**
   - FANNCortexM: An Open Source Toolkit for Deployment of Multi-layer Neural Networks on ARM Cortex-M Family Microcontrollers : Performance Analysis with Stress Detection
   - **CMSIS-NN**: Efficient Neural Network Kernels for Arm Cortex-M CPUs
   - NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems

2. Detail the level of formality with someone experienced in NN or machine learning.

Institute of embedded systems at TUHH

- **Test Optimization for Embedded Systems**
- Technology-Level Test Vector Generation
- **Verifying Artificial Neural Networks**
- **Assessing the Performance of Embedded Machine Learning**

Thank you!

Institute of embedded systems at TUHH

Test Optimization for Embedded Systems

Embedded systems are not directly visible to the end user. However, they must be extremely reliable to guarantee correct behavior of larger technical systems. For this reason elaborated test procedures are applied that successively validate each embedded system before shipping. **Individual hardware components, electrical interfaces and, finally, the functionality of the complete embedded system including software are incrementally tested.** The goal of this work is to work on testing data from a actually running testing process at an industrial partner's facilities in Hamburg.

Technology-Level Test Vector Generation

Test vectors for integrated circuits applied to each produced chip guarantee proper functionality and highest quality standards. More recent improvement in circuit technology and the use of advance approximate processing units makes a precise differentiation of good and bad chips on the logic level more and more difficult. The topic proposed here will consider low-level technology information for generating tests for advanced production processes and approximate hardware.

Verifying Artificial Neural Networks

Formally verifying the correctness of procedures and systems used in safety related areas is a must. Artificial neural networks have been proven very effective in solving many tasks in every day life. However, **methods deciding whether an artificial neural network never violates safety guarantees are only at their infancy. The goal of this work is to study the state-of-the-art in verifying correctness of artificial neural networks**.

Assessing the Performance of Embedded Machine Learning

Most suppliers of advanced embedded processing devices provide software libraries and hardware support to improve the performance of machine learning in embedded systems either for training classifiers, for applying learned classifiers or both. **But how efficient are these embedded platforms. The goal is to use and evaluate the performance of embedded machine learning platforms.**

Notes from papers

FANNCortexM: An Open Source Toolkit for Deployment of Multi-layer Neural Networks o ARM Cortex-M Family Microcontrollers [1]

FANN is a free open source neural network library that implements multi-layer ANNs in C. These can be ported and optimized in ARM Cortex-M architectures.

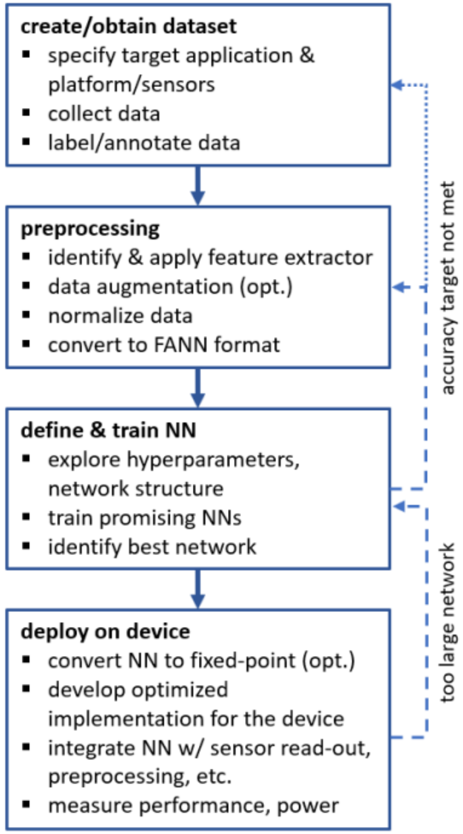
FANNCortexM highlights the importance of CMSIS (a vendor independent HAL for Cortex-M processors) by using optimized product functions that allows efficiency increments in execution time by ~30%.

Tested with a 6-input, 3-class perceptron with 2 hidden layers and evaluation of a network with 1000 nodes and 47900 weights (256kB).

Process flow: create/obtain dataset, preprocessing, define/train NN, deploy on device.

(Still analyzing)

Process flow according to [1]



create/obtain dataset
- specify target application & platform/sensors
- collect data
- label/annotate data

preprocessing
- identify & apply feature extractor
- data augmentation (opt.)
- normalize data
- convert to FANN format

define & train NN
- explore hyperparameters, network structure
- train promising NNs
- identify best network

deploy on device
- convert NN to fixed-point (opt.)
- develop optimized implementation for the device
- integrate NN w/ sensor read-out, preprocessing, etc.
- measure performance, power

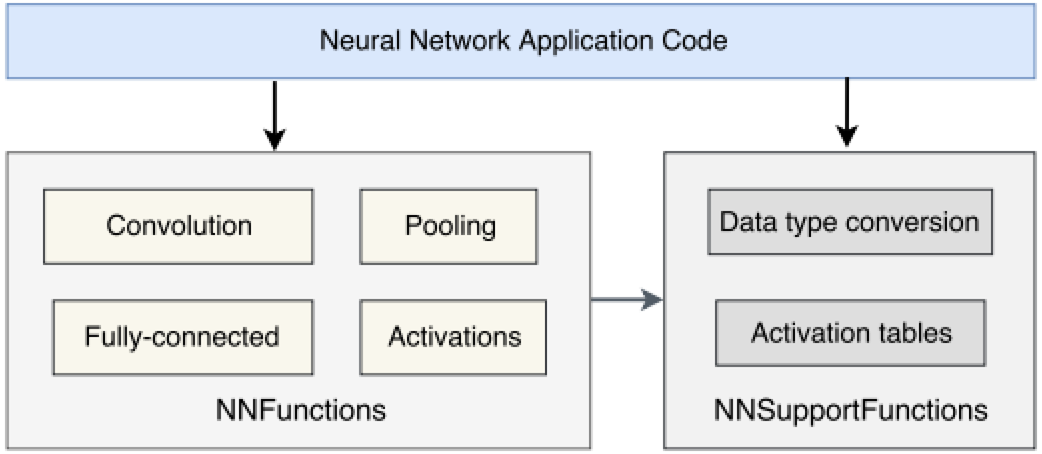
Proposal: details

Implementation of an ANN using FANNCortexM on a Axis Communication Hub to evaluate its performance and usage reliability within an EtherCAT Network.

- Use a pre-trained NN with multi-layer perceptrons instead of deep convolutional networks for an application based on a sensor that does not produce images.
- According to [1], it will be focused on define/train NN, deploy on device stages, using a given data set at an specific accuracy.
  - FANN conversion, hyperparametrization, identification of best network, FPU configuration, implementation to HW, measure performance and power.
- Current Axis Communication Hub integrates a STM32F446RTE MCU with ARM Cortex-M4 architecture, has FPU and it is within the high performance family ST processors. 512kB Flash/128+4KB RAM[2].

Notes from Papers

Neural Network Kernel functions structure



Notes from papers

NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems [3]

Motivated by the usage of DNNs on safety-critical cyber-physical systems (CPS)

Supports both sound and complete reachability algorithms for verifying safety and robustness properties of FFNN. (Perceptrons and deep convolution NN are FFN)

Uses set representations such as polyhedral, zonotopes and abstract domain representations. (to be reviewed)

Works on Matlab tool, taking translated NN models exported by Keras and Tensor using ONNX format and HyST transformation tool.

(Still analyzing)

Proposal: details

Review the possibility of using the NNV tool for models exported/transformed by the CMSIS-NN or FANN-CortexM frameworks.

Are the reachability analysis' results through NNV translatable to FFNN running on ARM-architectures.

Does the sensor readout functionality of the ACB fulfill the a-priori requirements for NNCS of the NNV tool? Such that the NNV works properly on current application?

Two possible applications: Force-torque sensor or speech recognition.

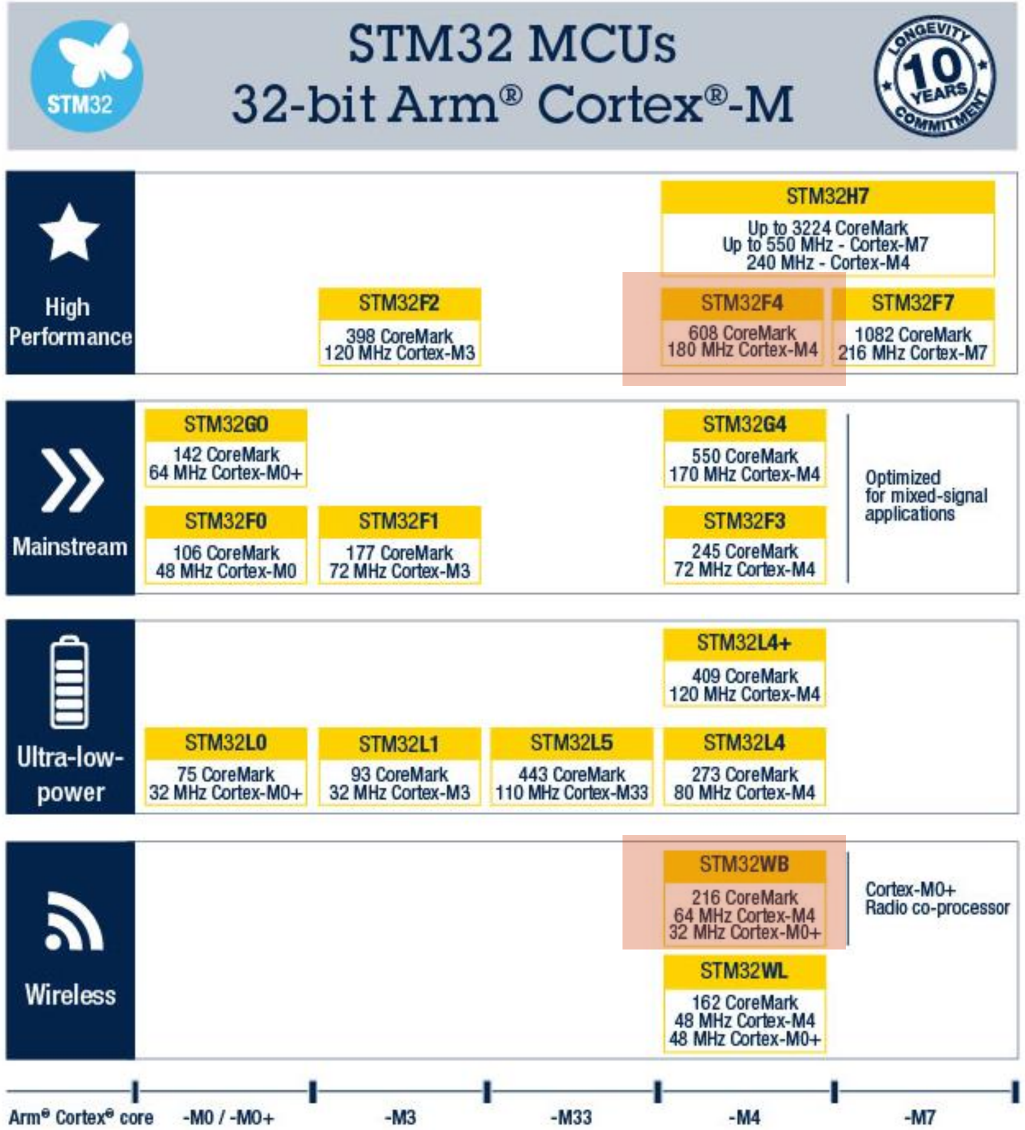
References

[1] M. Magno, L. Cavigelli, P. Mayer, F. v. Hagen and L. Benini, "FANNCortexM: An Open Source Toolkit for Deployment of Multi-layer Neural Networks on ARM Cortex-M Family Microcontrollers : Performance Analysis with Stress Detection," 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 2019, pp. 793-798, doi: 10.1109/WF-IoT.2019.8767290.

[2] J. C. Reyes, “Development of an embedded communication hub for sensor data acquisition in a robotic system”, September 2020, Research Project supervised by the Research Group smartPORT of the Technical University of Hamburg (TUHH)

[3] Tran, Dung & Yang, Xiaodong & Manzanas Lopez, Diego & Musau, Patrick & Nguyen, Luan & Xiang, Weiming & Bak, Stanley & Johnson, Taylor. (2020). NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems. 10.1007/978-3-030-53288-8_1.

ST Portfolio



STM32 MCUs
32-bit Arm® Cortex®-M

Arm® Cortex® core	-M0 / -M0+	-M3	-M33	-M4	-M7
High Performance				STM32H7 Up to 3224 CoreMark Up to 550 MHz - Cortex-M7 240 MHz - Cortex-M4	
		STM32F2 398 CoreMark 120 MHz Cortex-M3		STM32F4 608 CoreMark 180 MHz Cortex-M4	STM32F7 1082 CoreMark 216 MHz Cortex-M7
Mainstream	STM32G0 142 CoreMark 64 MHz Cortex-M0+			STM32G4 550 CoreMark 170 MHz Cortex-M4	Optimized for mixed-signal applications
	STM32F0 106 CoreMark 48 MHz Cortex-M0	STM32F1 177 CoreMark 72 MHz Cortex-M3		STM32F3 245 CoreMark 72 MHz Cortex-M4	
Ultra-low-power				STM32L4+ 409 CoreMark 120 MHz Cortex-M4	
	STM32L0 75 CoreMark 32 MHz Cortex-M0+	STM32L1 93 CoreMark 32 MHz Cortex-M3	STM32L5 443 CoreMark 110 MHz Cortex-M33	STM32L4 273 CoreMark 80 MHz Cortex-M4	
Wireless				STM32WB 216 CoreMark 64 MHz Cortex-M4 32 MHz Cortex-M0+	Cortex-M0+ Radio co-processor
				STM32WL 162 CoreMark 48 MHz Cortex-M4 48 MHz Cortex-M0+	