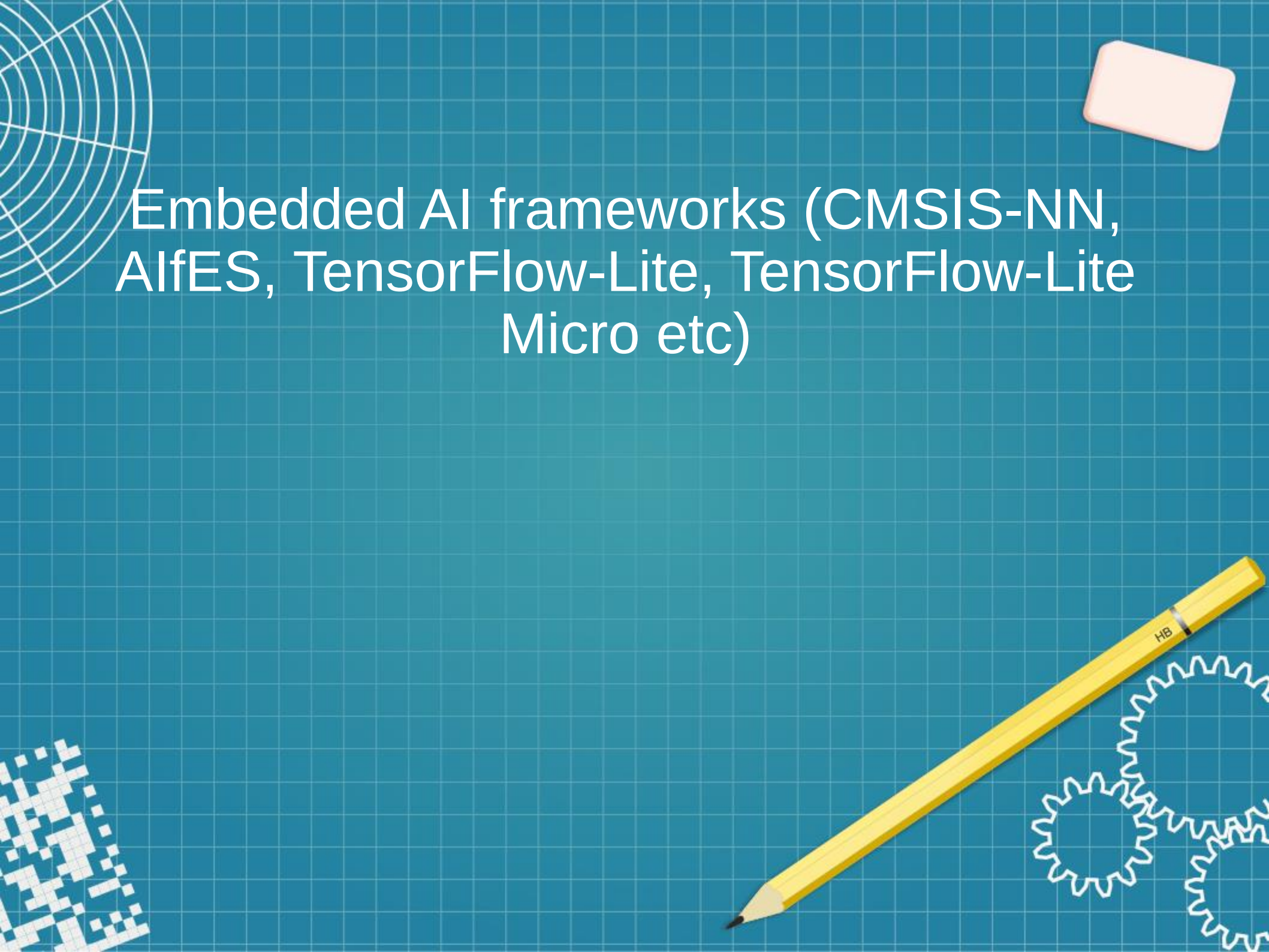
Embedded AI frameworks (CMSIS-NN, AIfES, TensorFlow-Lite, TensorFlow-Lite Micro etc)



Vision & Mission

Vision:

To make Intelligent and self-learning embedded systems.

Mission:

Seamless ML integration at the data source, be it a sensor, machine, or system, independent of hardware.

Purpose of TinyML or Embedded-AI

Embedded-AI as a solution for resource-limited systems:

Decentralized, highly integrated AI at the point of data generation (Sensor, component, product, device)

has the following advantages:

- Fast processing
- Increased security
- Increased reliability
- Saving resources
- Saving energy
- Personalizable AI

Tensorflowlite

- TensorFlow Lite is optimized for mobile and edge devices to ensure efficient execution of machine learning models on devices with limited processing power, memory, and storage.
- TensorFlow Lite supports the conversion of TensorFlow models into a format that can be efficiently executed on mobile and edge devices. This conversion process helps in reducing the model size and making it compatible with the target platform.
- Quantization is a technique used in TensorFlow Lite to reduce the precision of the model's weights and activations. This results in smaller model sizes and faster inference, making it well-suited for edge devices.
- TensorFlow Lite uses an interpreter to run the machine learning models on devices. The interpreter is designed to be lightweight and efficient, allowing for quick and real-time inference on edge devices.
- TensorFlow Lite supports hardware acceleration to leverage the specialized processing units available on some devices, such as GPUs (Graphics Processing Units) and TPUs (Tensor Processing Units). This enhances the performance of machine learning applications.
- TensorFlow Lite maintains compatibility with TensorFlow, making it easy for developers to develop and train models using the TensorFlow framework and then deploy them on mobile and edge devices using TensorFlow Lite.

TensorFlow Lite - a primer

Mobile and embedded devices

Definition of Embedded Devices:

An embedded device is an object that contains a special-purpose computing system. The system, which is completely enclosed by the object, may or may not be able to connect to the Internet.

Generally, an embedded device's operating system will only run a single application which helps the device to do its job. Examples of embedded devices include dishwashers, banking ATM machines, routers, point of sale terminals (POS terminals) and cell phones. Devices that can connect to the Internet are called smart or intelligent. If an embedded device can not connect to the Internet, it is called dumb.

What are the limitations of embedded devices

Because embedded systems have limited computing resources and strict power requirements, writing software for embedded devices is a very specialized field that requires knowledge of both hardware components and programming.

What are the main objectives we must keep in mind

- Model Size:We need a small model (file) size to reduce possible download time and RAM usage.
- Latency, Battery and heat:We need to reduce amount of computations needed for inference
- Other constraints:We may have additional constraints. For example no UI or user interaction. Or slow computational power.

Process to use a model on the edge

- Train and save a model (development machine)
- Convert the model (development machine)
- Copy the converted model on the device
- Run inference with the TF Lite interpreter

Main reason to convert a model

The main reasons to convert the model are

- Make it smaller (smaller memory footprint)
- Make inference more efficient
- Require less memory access
- Use less energy in inference

To achieve those goals the main component is what is called **QUANTIZATION.**

Quantization

- Quantization is a technique used in machine learning to reduce the precision of the numerical values in a model, typically from 32-bit floating-point numbers to lower bit-width representations like 8-bit integers.
- This process helps in reducing the model size, making it more memory-efficient, and accelerating inference on hardware that supports lower precision computations.
- In the context of TensorFlow Lite (TFLite), quantization is an important optimization technique to deploy machine learning models on resource-constrained devices.

There two types of quantization

- Post-training quantization - more simple and easier to implement and in most cases extremely efficient
- Training-aware quantization - more complex that require the re-writing of the computational graph (see for example https://github.com/tensorflow/tensorflow/tree/r1.13/tensorflow/contrib/quantize)

Quantization - the formula

For weight quantization in neural networks, a common formula is:

- Quantized Value=Round(Original Value/Scale Factor)

Here:

- Original Value: The original weight value in floating-point precision.
- Scale Factor: A scaling factor that helps map the original floating-point values to a quantized integer representation.
- Round: This function rounds the result to the nearest integer. It is necessary because quantized values are integers.

TensorFlow Lite for Microcontrollers (TFLu)

- Version of TensorFlow Lite designed to execute neural networks on microcontrollers, starting at only a few kB of memory.
- Designed to be portable even to 'bare metal' systems
- The core runtime is ~20kB.
- Generate multiple projects, for example MbedOS and Arduino.
- Over 50 operators supported currently. Growing quickly.
- Many integrated operator optimizations.

Examples/demos

- Micro speech: Detects simple commands such as yes, no and silence.
- Person detection: Detects whether a person is in the room or not.
- Magic wand demo for image recognition etc.

CMSIS-NN

- Arm CMSIS-NN library, version CMSIS 5.7.0.
- The Cortex-M7 processor is found in a range of solutions.
- Its Increased compute performance in the smallest Cortex-M-based devices is enabling more data processing to move to the edge.
- The CMSIS-NN library is designed to optimize the performance of neural network algorithms on resource-constrained microcontroller platforms.
- It provides support for fixed-point arithmetic, which is often preferred in embedded systems due to its reduced computational complexity compared to floating-point arithmetic.
- CMSIS-NN is designed to minimize memory requirements, making it suitable for microcontroller applications where resources such as RAM and flash memory are limited.
- Some implementations of CMSIS-NN may leverage hardware features available on certain Cortex-M processors to further enhance performance.

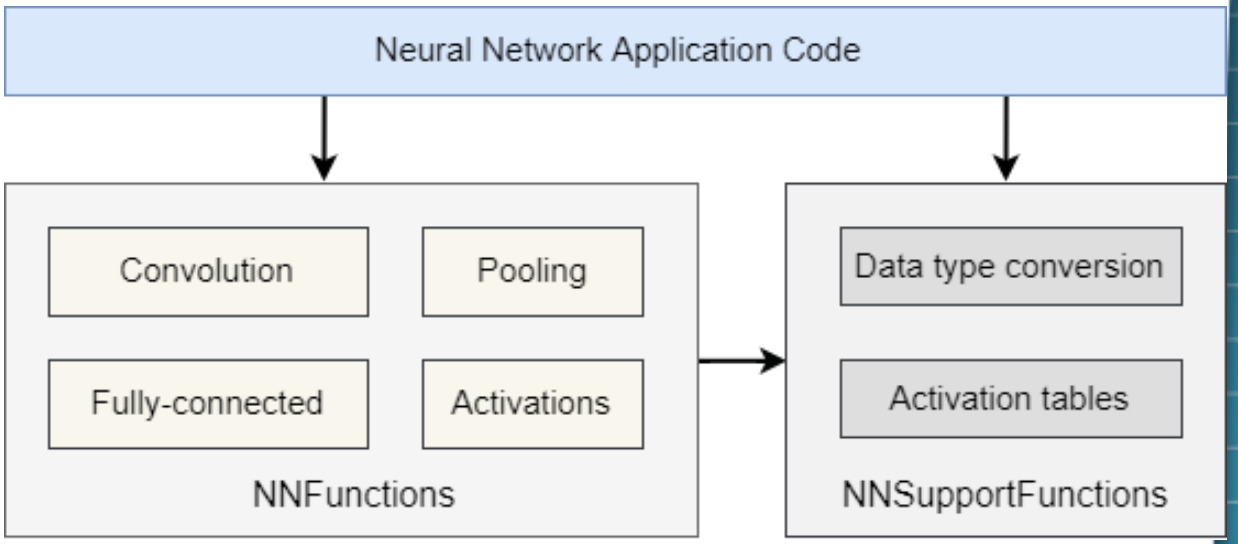
CMSIS-NN

- CMSIS-NN provides a broad set of neural network kernels for Cortex-M devices.
- Optimized implementations for different Cortex-M capabilities (SIMD, FPU, MVE).
- Arm Compiler 6 and on Arm GNU Toolchain support.
- Follows int8 and int16 quantization specification of TensorFlow Lite for Microcontrollers.
- Widely adopted in the industry.

CMSIS-NN

Supported functions

- Convolution Functions
- Activation Functions
- Fully-connected Layer Functions
- SVDF Layer Functions
- Pooling Functions
- Softmax Functions
- Basic math Functions



AIfES

- Standalone, open-source AI framework.
- High-efficiency implementation in C.
- Enables training and execution of ML algorithms on small microcontrollers.
- Developed by Fraunhofer Institute for Microelectronic Circuits and Systems IMS.
- IfES thus runs on almost any hardware from 8-bit microcontrollers over Raspberry PI to smartphones or PCs
- Not only inference of FNN is possible, but also training directly in the device. Furthermore, compatibility to other AI software frameworks such as Keras or TensorFlow is also given.

AIfES features

- Open-source library for standalone AI development.
- No conversions needed, seamlessly integrates into any C or C++ IDE.
- Develop artificial neural networks (ANN) directly on target hardware.
- Supports inference and training of ANNs.
- Programmed in ISO C and GCC compatible.
- Compatible with a broad range of hardware, from 8-Bit microcontrollers to Pcs.
- Allows multiple ANNs on one system and runtime reconfiguration.
- Import ANNs from other frameworks is supported.

AlfES for Arduino

- AlfES and Arduino partnership
- Compatible with Arduino IDE
- Runs on various Arduino or Arduino compatible boards
- Published under GNU GPL V3 license

AIfES licensing and partners

- Dual License Model: GNU GPL V3 for private or free open-source use.
- Commercial use possible under a paid license agreement.
- Partnerships with Arduino, Arm AI Ecosystem, Open Roberta Lab.

Function overview

Feedforward neural network Inference

- Float
- Freely configurable (inputs, hidden layer, outputs)
- Many activation functions - Sigmoid, softsign, linear, RELU, Leaky RELU, softmax, tanh, ELU

Feedforward neural network Training

- Full SGD and ADAM algorithm
- Training Types-Online, Batch, Minibatch
- Various loss functions-mean squared error (MSE), cross-entropy

AIfES Updates and Future Developments

- AIfES-Express API for simplified integration
- Fixpoint calculation with quantization of weights
- Storage of weights in flash memory
- Ongoing development of CNN and reinforcement learning support