Analysis of Frameworks for Developing Artificial Intelligence Models on Low-Power Microcontrollers in Wireless Sensor Networks: Capabilities and Challenges

Krzysztof Turchan, Krzysztof Piotrowski
IHP - Leibniz Institute for High Performance Microelectronics
Frankfurt (Oder), Germany
{turchan, piotrowski}@ihp-microelectronics.com

***Abstract*—This paper presents a comprehensive analysis of various frameworks that enable the development of artificial intelligence (AI) models on low-power microcontrollers used in the context of Wireless Sensor Networks (WSN). Edge computing is becoming very popular, and microcontrollers play a crucial role in implementing intelligent systems with limited resources. This study focuses on evaluating the available frameworks regarding their capabilities and limitations in programming AI applications on microcontrollers. The analyzed frameworks differ in terms of complexity and intended use—some are suitable for advanced engineering projects, while others are ideal for educational purposes, such as teaching students how to program embedded AI applications using popular IoT microcontrollers. Additionally, the paper discusses challenges such as memory constraints, code optimization requirements, and difficulties in implementing advanced algorithms. The goal is to provide practical guidance for engineers and researchers developing intelligent applications in resource-constrained environments.**

***Index Terms*—Wireless Sensor Network, Artificial Intelligence, AI Frameworks, AI Models for WSN**

I. Introduction

WSN nodes are devices whose task is to collect data from the environment. In a basic version of a WSN, this data is transmitted single- or multi-hop to a base node and placed in databases for further processing. This generates a lot of network traffic and often a lot of overall energy consumption. In contrast, it is very often the case that only the information resulting from this data is important. Using the IF/ELSE approach to optimize the amount of data sent makes sense only with a small number of variables in the code. The problem occurs with a larger number.

Artificial intelligence algorithms might be helpful, because their task is to extract information from the data in a non-linear way. These, on the other hand, are very resource-intensive, which in the case of WSN nodes is a strong limitation. Increasing the computational power of the nodes is reflected in increased power consumption, which, with battery power, is the most essential element. Thus, the approach to running these algorithms must move in the direction of optimizing both the algorithms themselves and the data that is calculated on the node.

Writing algorithms from scratch is not a trivial task and is it worth using frameworks that are dedicated for it. A framework is a structure or set of libraries, tools and conventions that facilitate the development of applications or information systems. There is a group of frameworks designed to facilitate the development of AI applications for embedded systems. Such a framework must be able to train an artificial intelligence model from sensor data, and allow this model to be used on nodes with limited resources, at least in terms of inference.

The following sections cover the methodologies used to examine the frameworks, followed by a detailed description of a few selected ones. The analysis presented here was done in theoretical terms, as a state of the art for the concept described in the section *Distributed Data-Centric Application Logic Layer*. During the analysis, the most important features of the frameworks from the WSN point of view were highlighted.

II. Methodology

An AI framework for WSN nodes must meet some, or preferably all, of the criteria outlined below:

It must support model learning for easy and intermediate AI algorithms such as decision trees, linear regression, logistic regression, simple K-NN tasks, Kalman filters, simple neural networks, etc.

Additionally, a major advantage of the framework is that it allows for further training of the model while the node is operating, enabling it to adapt to current conditions.

It must support the C programming language, at least for the inference phase, since it is still the most popular choice in embedded programming.

Speaking of compatibility, the universality of the framework is a very big advantage. The functions it generates cannot contain elements specific to any of the microcontroller families.

However, the most important is all kinds of optimization. It must optimize the code, models, and data to maximize the performance of the microcontroller during operation, while maintaining the best possible results during inference.

III. Evaluation of Frameworks

This section contains a description of the frameworks that were taken for analysis.

A. microTensor (uTensor)

The first considered framework is uTensor [1]. It was designed to facilitate the implementation of machine learning (ML) models on ARM microcontrollers. It is a limitation from the perspective of a generic WSN node but has other features important for an ideal framework. It provides a runtime library coupled with an offline toolchain, which together convert and optimize ML models, allowing them to be used on microcontroller. Unfortunately, only for the purpose of inference.

What definitely needs to be emphasized is that uTensor uses many different optimization methods. It supports post-training quantization [2], reducing model size and computational load by converting 32-bit floating-point operations to 8-bit integers, which are more suitable for microcontroller operations.

The framework implements optimized kernels for common operations, such as convolutions and matrix multiplications. This accelerates the execution of neural networks by utilizing SIMD (Single Instruction, Multiple Data) instructions.

uTensor features a highly optimized memory allocator that efficiently manages the limited RAM available on microcontrollers. It uses techniques such as memory pooling and static memory planning to minimize fragmentation and ensure deterministic memory usage. And that feature is really interesting and should be considered in every single framework.

B. TensorFlow Lite for Microcontrollers

TensorFlow Lite for Microcontrollers enables developers to deploy ML models on resource-constrained embedded devices. It's a lightweight framework, that is built on top of TensorFlow. It allows inference at the node level, unfortunately without the ability to further train the model. It supports most of the basic AI models described in the methodology section, which is a very big advantage.

The framework is compatible with a wide range of microcontrollers [3]. It leverages techniques like quantization and kernel acceleration for efficient inference. It integrates with TensorFlow tools and models for improved development. Of course it requires a deeper understanding of TensorFlow concepts by that. It has limited programming language support. It mainly supports C++ for model inference and Python for model preparation. Since the code generated by TFLM is in C++, using it on many WSN nodes is either impossible or requires significant effort.

Overall, TensorFlow Lite for Microcontrollers is a powerful tool for developing intelligent edge devices. Its strengths lie in hardware support, performance optimization, and TensorFlow integration. However, the learning curve, language support limitations, and tool integration should be considered when selecting this framework.

C. Edge Impulse

Edge Impulse is distinguished from the rest of the frameworks by the fact that it is a cloud-based development platform. It provides a user-friendly interface, powerful tools, and a supportive community, making it an attractive choice for developers of all skill levels [4].

Another thing that sets this framework apart is graphic user interface (GUI) and intuitive workflow make it easy to learn and use, even for beginners. It offers a complete set of tools for data collection, preprocessing, model training, and deployment. It supports a wide range of edge devices and hardware platforms. It optimizes models for performance and efficiency on resource-constrained devices. Benefits from a large and active community with forums, tutorials, and documentation.

Cloud dependency is both advantage and disadvantage. Offers less flexibility for experienced developers who prefer more control over the development process, and of course, it doesn't work without an internet connection. One more disadvantage is pricing - free plan has limitations, and paid plans may be required for advanced features and larger projects.

Overall, Edge Impulse is a good platform for developing and deploying ML models for edge devices. Its ease of use, comprehensive toolset, and strong community make it a compelling choice for a wide range of users. However, the cloud dependency, limited customization, and pricing considerations should be factored into the decision-making process.

D. AIfES

AIfES (Artificial Intelligence for Embedded Systems) is another embedded AI framework designed for running ML models on microcontrollers and other resource-constrained devices. The difference is that AIfES allows the model to be retrained on the device [5]. Which greatly distinguishes it in this group of frameworks.

AIfES is highly portable C-based library that enables the development and deployment of neural networks on 8-bit, 16-bit, and 32-bit microcontrollers. It is designed to be lightweight, with a minimal memory footprint, making it suitable for even the most resource-constrained environments. The framework supports various types of neural networks, including multilayer perceptrons (MLPs). The framework is implemented in C, which ensures high portability across different microcontroller architectures.

As mentioned AIfES allows for both the training and inference of models directly on microcontrollers. This is particularly valuable in scenarios where real-time model updates are needed, such as WSN, where the environment changes throughout the year. The framework includes various training algorithms, including backpropagation, allowing for on-device learning. It handles every single simple models like MLPs, which are appropriate for tasks such as classification, regression, and sensor data processing.

The framework uses static memory allocation to avoid the overhead of dynamic memory management, which is crucial in microcontroller environments where RAM is extremely limited. It also optimizes memory usage by reusing buffers where

TABLE I
EVALUATION OF FRAMEWORKS COMPATIBILITIES

Feature	microTensor	TensorFlow Lite for Microcontrollers	AIfES	Edge Impulse
Ease of Use	Moderate: familiarity with c++ and TensorFlow	Moderate: Familiarity with TensorFlow	Difficult: Deep understanding of neural networks	Easy: User-friendly GUI and intuitive workflow
Programming Languages	C++ (Inference) and Python (model learning and converting)	C++ (Inference) and Python (model learning and converting)	C	No coding required
On-Device Training Support	Inference only	Inference only	Inference and on-device training	Inference only
Hardware Support	ARM Cortex-M based microcontrollers	ARM Cortex-M based microcontrollers, Android	Any microcontroller that supports the C programming language	Wide range of edge devices
Support for Various AI Models	Supports various model architectures, including CNNs	Supports various model architectures, including CNNs	Almost every basic and intermediate AI algorithm	Supports a wide range of model types through model conversion
Performance and Optimization	Quantization, Pruning [6] and memory allocation techniques	Quantization, Pruning, Integer-only models	Quantization, Pruning and memory allocation techniques	Quantization, Transfer Learning, Pruning, Knowledge distillation

possible. And additionally it is designed to minimize power consumption, making it suitable for battery-operated WSN nodes. The framework can run on devices with limited power budgets, ensuring that ML tasks do not excessively drain the battery.

In summary, AIfES meets all the requirements of a good AI framework for WSN nodes, becoming an ideal base for the concept described in the next section.

IV. Distributed Data-centric Application Logic Layer

This section provides a brief overview of the use of AI framework in existing use case scenario which is Distributed Data-centric Applicatoin Logic Layer with tinyDSM.

Finding a suitable Framework was needed to extend the distributed application logic layer with artificial intelligence algorithms.

To properly introduce the topic of this layer, it's essential to first discuss the fundamental component on which it is based: tinyDSM. tinyDSM is a type of Middleware designed to facilitate cooperation between nodes in a WSN. It operates using Services, Adapters, and Data Interfaces [7].

Services are software modules connected to a Data Interface to perform specific data processing tasks. The term "service" is broad, and its specific role depends on the project in which tinyDSM is implemented. Another type of software module is the Adapter. Adapters are responsible for translating data (e.g., from sensors) into a format that can be understood by the Data Interface and, subsequently, by tinyDSM. The Data Interface itself serves as the communication medium, transmitting data within tinyDSM.

Nodes utilizing this solution can ensure reliable data transfer and perform distributed computations, including those based on artificial intelligence algorithms.

The layer described here consists of tinyDSM services that operate on the nodes, enabling them to collaborate in performing network-wide calculations. These calculations can range from tasks like computing average sensor values to determining averages over larger areas, such as on repeaters. By combining distributed logic with AI algorithms, it becomes feasible to develop WSNs that behave like neural networks. This is the primary aim of the research: to connect small, resource-constrained nodes into a computationally efficient network enhanced with AI.

V. Conclusion

Each of the presented frameworks offers different possibilities. The use of each must be justified by different needs, mainly project needs. It may be both a project created by a embedded programming enthusiast, or by professional. For the first purpose, the free version of Edge Impulse should be considered. However, for more professional solutions, the AIfES framework is highly recommended. AIfES supports ongoing model training while nodes are in operation, allowing the network to adapt to changing conditions in real-time. Additionally, its strong compatibility with C programming and optimization for low-power microcontrollers ensures efficient performance in resource-constrained environments typical of WSNs.

References

[1] uTensor website, https://github.com/uTensor/website/blob/master/docs.md, last accessed 22.07.2024.
[2] M, Pietrołaj, M, Blok, "Resource constrained neural network training, Dental science reports," 2024.
[3] TensorFlow lite for microcontrollers, https://www.tensorflow.org/lite/microcontrollers, last accessed 22.07.2024.
[4] Edge Impulse Docs, https://docs.edgeimpulse.com/docs, last accessed 22.07.2024.
[5] L. Wulfert et al., "AIfES: A Next-Generation Edge AI Framework", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 46, no. 6, pp. 4519–4533, 2024.
[6] J. Wang, "Research on pruning optimization techniques for neural networks, Applied and Computational Engineering," 2023.
[7] K. Piotrowski, P. Langendoerfer, and S. Peter, "tinyDSM: A highly reliable cooperative data storage for Wireless Sensor Networks", in 2009 International Symposium on Collaborative Technologies and Systems, 2009, pp. 225–232.