Machine Learning on Standard Embedded Device

Umapriya Selvam
Electronics & Communication Engineering,
Coimbatore Institute of Technology, Coimbatore, India

P. Muthu Subramanian
Assistant Professor
Electronics & Communication Engineering,
Coimbatore Institute of Technology, Coimbatore, India

A. Rajeswari
Professor
Electronics & Communication Engineering,
Coimbatore Institute of Technology, Coimbatore, India

ABSTRACT
Developers of ARM microcontrollers now have access to the first neural network software development tools, making machine learning in embedded systems a possibility. This study examines the application of one such tool, the STM Cube AI, on popular ARM Cortex-M microcontrollers. It evaluates and contrasts its performance with that of two others widely employed supervised machine learning (ML) algorithms, namely Support Vector Machines (SVM) and k-Nearest Neighbors (k-NN). The outcomes of three datasets demonstrate that X-Cube-AI consistently delivers good performance despite the shortcomings of the embedded platform. Popular desktop programs like TensorFlow and Keras are seamlessly incorporated into the workflow.

General Terms
k- Nearest Neighbors, X-Cube-AI, ARM Cortex-M

Keywords
Machine learning, Artificial neural networks, Microcontrollers, Edge Computing

1. INTRODUCTION
Numerous new field applications are being made possible by the Internet of Things (IoT). The enormous amount of data produced by IoT sensors is increasingly being processed close to the source, at the edge, which limits privacy concerns while reducing latencies, bandwidth, and overhead of the cloud and remote units [1]. The edge computing paradigm is an addition to the widely recognized cloud computing model. Within this framework, artificial intelligence, particularly Machine Learning (ML), has gained significance. ARM recently unveiled the Project Trillium ML platform, which encompasses machine learning and object detection intellectual property (IP) designed for highly advanced smartphones.However, it is probable that widespread adoption will primarily occur on already popular platforms. For instance, in 2018, ARM developed CMSIS NN, an open source library of optimized kernels, with the objective of maximizing Neural Network (NN) performance specifically on Cortex-M processors. These processors are widely utilized in the field, making them the preferred choice for maximizing NN capabilities. Google launched TensorFlow Lite for ARM 64 microcontrollers, with a focus on NNs . STM has recently introduced the STM Cube AI extension kit, which is designed specifically for 32-bit microcontrollers. This kit aims to provide enhanced capabilities for integrating artificial intelligence functionalities into microcontroller-based systems.However, there are significantly fewer articles discussing experiences with ML on the edge than there are for desktop/cloud computing. This is made worse by the dearth of publicly accessible IoT datasets, which hinders the
creation of research initiatives. The purpose of this paper is to investigate and evaluate the performance of one of the aforementioned N N libraries, X Cube AI, on popular ARM Cortex-M microcontrollers while also taking into account two additional widely used supervised machine learning (ML) algorithms, Support Vector Machine (SVM) and k-Nearest Neighbors (k-NN). Since the training step is extremely labor-intensive [2] the study is initially focused on classification because it frequently calls for human supervision, which is easier to carry out in the cloud.
available, try the font named Computer Modern Roman. On a Macintosh, use the font named Times. Right margins should be justified, not ragged.

2. SURVEY
Numerous studies are being conducted to integrate ANNs into autonomous devices, addressing difficulties with accuracy, resource utilization, and energy efficiency. A thorough examination of the current work in this area is given in [3].

Challenges of deploying neural networks on microcontrollers with limited memory, compute, and power budgets [4].The authors introduce the CMSIS-NN library of effective software kernels, which allows the implementation of NNs on Cortex-M cores. They also discuss techniques for NN algorithm development to build compact models appropriate for resource-constrained systems using the example of keyword spotting. The integration of a low resolution thermal imaging camera using cutting edge feature extraction techniques like convolutional layers is demonstrated in [6] as a unique system. The research demonstrates the capability of changing the classification process to a resource-constrained platform without appreciably decreasing performance by analyzing information on a 32-bit low power microcontroller. Using 6 kB of RAM, they attain a 77 percent accuracy. In addition to other techniques like genetic and reinforcement learning, it implements a NN for classification.

3. IMPLEMENTATION
As many have already said, NNs have become more popular in the realm of embedded systems. One of the recently published libraries described above, the STM Cube AI expansion package, which may be used with the STM32CubeMX configuration tool, is the subject of our investigation in particular. Pre-trained neural networks are automatically converted by the programme, and the generated optimized library is integrated into the user's project. The method we are familiar with entails building a NN in Python using the Tensorflow package and Keras as a framework on a computer. In order to speed up convergence,the vectors are normalized. When the developer discovers a NN configuration that, in testing conducted on a PC, provides sufficient accuracy, the model is stored in a HDF5 file and imported by CubeMX. After estimating the memory footprint (including Flash and RAM), the CubeMX ``Analyze'' function then proposes a list of potential target microcontrollers. A new project, which includes the AI and CUBE AI packs, can be launched once the

target has been chosen (or the developers have verified the suitability of the current target).

Then, using the Multiply and Accumulate Operation to estimate complexity, CubeMX enables executing a validation both on the desktop and on the target. By utilizing the "network" package, constructing the system using the collected weights, establishing the input and output tensors, and doing the prediction, the target C programme may be written in a few lines of code. Two algorithms were used as a comparative basis,

3.1 K-nearest neighbors

In k-NN, the entire training set is recorded instead of any model being learned. Using majority voting and the Euclidean distance criterion, we completely rewrote the system in C.

3.2 Support Vector Machine

On a computer,the SVM was **trained** using the Sklearn Python framework using a linear kernel and a cross-validation model. Both the SVM method and gpu acceleration are not supported by sklearn, and multi core architectures cannot be utilized by the SVM method. This is a drawback of our strategy because the lengthy training periods precluded us from thoroughly examining the choices (kernels). Executing the prediction z = x*y+w on the target is all that is required for implementation, where z and y represent the inputs and output, x the support vectors, and w the bias.

4. ANALYSIS

F446RE and F746 from STM, two well-known ARM microcontrollers, were used to carry out the experimental analysis. The former is a member of the popular Cortex-M4 family, while the latter is a member of the high-performance M7. Results are typically only presented for the F4 scenario in Tables 1-4, while Table 3 expressly takes into account the F7 condition. In each case, Prior to deploying the classifiers on the target machine and making the required adjustments, notably in terms of performance, the classifiers were first developed on a PC.

Sonar (207 samples x 58 features) , the UCI Heart illnesses dataset on Kaggle (301 x 11),were the two binary classification datasets used. In accordance with the intended execution platform, each dataset is converted to a float32. Table 1 shows the Sonar dataset's data for a NN having 2 hidden thick layers. (30 and 20 after a first ReLU thick layer with 99 nodes, tan h neurons each). A more complicated network results in a 243 kB Flash footprint and an accuracy of 83 %. The ideal k for k-NN is 1. Accuracy for all classifiers is equivalent to that of an i4-core computer.

Table.1 Sonar dataset classification results

Classifier	**50 features**		
	Flash	Time	Accuracy
K NN	45kB	24ms	79%
SVM	249B	< 1ms	83.4%
NN	49kB	< 1ms	91%

The presentation of the k-NN (for 11 features k=13) required feature selection (Orthogonal Matching Pursuit algorithm), which is shown in Heart infection dataset classification Table 2. A 3-layer NN with 40, 20, and 1 node was employed, and output was nonlinear for all nodes. Despite utilizing the same code and dataset for every classifier, precision is identical to that on an i4 core PC, with the exception of the SVM.

Table.2 Heart infection dataset classification results

CLASSIFIER	**11 FEATURES**		
	Flash	**Time**	**Accuracy**
K NN	11kB	32ms	71%
SVM	28B	< 1ms	86.4%
NN	1.8kB	< 1ms	92.3%

5. CONCLUSION

Developers have access to the initial NN firmware development tools already, making ML in embedded devices a possibility. Using two different algorithms and three various datasets for analysis, we discovered that the NNs integrated by the STM Cube AI library consistently deliver good performance despite the embedded platform's constraints. Common desktop tools like Tensorflow and Keras are well integrated into the workflow. Additionally, SVM has a compact footprint and performs admirably.

Comparatively to NNs, however, its growth is not as effectively supported by tools. It is well known that as training set size increases, k-NN highly reliable to deteriorate [7].Openly accessible IoT datasets are currently lacking, which would help researchers and practitioners learn about various application domains.A more in-depth investigation in the future using more NN types and more pertinent datasets, with a focus on the space-time tradeoff. is to be conducted because they don't require human data processing during the training stage, uncontrolled classification algorithms are much more suitable for field deployment, and will also be fascinating to explore in terms of performance and application. Finally, spreading embedded ML compute is anticipated to emerge as a significant architectural problem in the coming years given the constrained facilities of the edge.

6. REFERENCES

[1] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu (2016) “Edge Computing: Vision and Challenges”, in IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646.

[2] A. Parodi, F. Bellotti, R. Berta, A. De Gloria (2018), “Developing a Machine Learning Library for Microcontrollers” Springer Lecture Notes in Electrical Engineering, vol 550.

[3] L. Andrade, A. Prost-Boucle, and F. Pétrot, Overview of the state of the art in embedded machine learning, (2018), “ Design, Automation & Test” in Europe Conference & Exhibition (DATE), Dresden, pp. 1033-1038.

[4] L. Lai and N. Suda (2018), "Enabling Deep Learning at the LoT Edge," in IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Diego, CA, pp. 1-6.

[5] G. Cerutti, R. Prasad and E. Farella (2019), "Convolutional Neural Network on Embedded Platform for People Presence Detection in Low Resolution Thermal Images," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton,

United Kingdom, 2019, pp. 7610-7614.

[6] M. J. Islam, Q. M. J. Wu, M. Ahmadi, and M. A. Sid-Ahmed, (2007), "Investigating the Performance of Naive-Bayes Classifiers and K- Nearest Neighbor Classifiers", Int.l Conf. on Convergence Information Technology , Gyeongju, pp. 1541-1546.