2023 NDIA MICHIGAN CHAPTER
GROUND VEHICLE SYSTEMS ENGINEERING
AND TECHNOLOGY SYMPOSIUM
<AUTONOMY, ARTIFICIAL INTELLIGENCE, & ROBOTICS (AAIR)> TECHNICAL SESSION
AUGUST 15-17, 2023 - NOVI, MICHIGAN

A LOW POWER AND HIGH PERFORMANCE SOFTWARE APPROACH TO ARTIFICIAL INTELLIGENCE ON-BOARD

Pablo Ghiglino, PhD[1], **Mandar Harshe, PhD**[2]

[1]Klepsydra Technologies, Zurich, Switzerland
[2]Klepsydra Technologies, Zurich, Switzerland

ABSTRACT

New generations of ground vehicles are required to perform tasks with an increased level of autonomy. Autonomous navigation and Artificial Intelligence on the edge are growing fields that require more sensors and more computational power to perform these missions. Furthermore, new sensors in the market produce better quality data at higher rates while new processors can increase substantially the computational power. Therefore, near-future ground vehicles will be equipped with large number of sensors that will produce data at rates that has not been seen before, while at the same time, data processing power will be significantly increased. This new scenario of advanced ground vehicles applications and increase in data amount and processing power, has brought new challenges with it: low determinism, excessive power needs, data losses and large response latency. In this article, a novel approach to on-board artificial intelligence (AI) is presented that is based on state-of-the-art academic research of the well known technique of data pipeline. Algorithm pipelining has seen a resurgence in the high performance computing work due its low power use and high throughput capabilities. The approach presented here provides a very sophisticated threading model combination of pipeline and parallelization techniques applied to deep neural networks (DNN), making these type of AI applications much more efficient and reliable. This new approach has been validated with several DNN models and different computer architectures. The results show that the data processing rate and power saving of the applications increase substantially with respect to standard AI solutions, enabling real AI on harsh environments like ground vehicle deployment.

Citation: P. Ghiglino, M. Harshe, "A Low Power And High Performance Software Approach to Artificial Intelligence On-Board," In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 15-17, 2023.

1. INTRODUCTION

It is common ground that embedded systems have evolved hugely in the last decade [1]. New generations of autonomous embedded systems are required to perform more and faster on-board data processing. Sensors, embedded processors, and hardware in general have hugely evolved in the last decade, equipping embedded systems with large number of sensors that will produce data at rates that has not been seen before while simultaneously having computing power capable of large data processing [2], [3]. This evolution in the hardware together with these new challenging AI requirement suppose an enormous challenge for on-board software engineering.

1.1 AI onboard Ground Vehicles

AI is used in ground vehicles for a variety of purposes, including:

- Autonomous navigation: AI can help military ground vehicles navigate autonomously, using sensors and algorithms to detect and avoid obstacles and make real-time decisions about where to go.
- Target detection and identification: AI can be used to identify potential targets on the battlefield, including enemy troops, vehicles, and equipment, by analyzing data from sensors and other sources.
- Predictive maintenance: AI can be used to monitor the health and performance of military ground vehicles, predicting when parts may fail and scheduling maintenance before a breakdown occurs.
- Predictive maintenance: AI can be used to monitor the health and performance of military ground vehicles, predicting when parts may fail and scheduling maintenance before a breakdown occurs.
- Driver assistance: AI can assist human drivers in controlling the vehicle, providing real-time feedback and alerts to help prevent accidents and improve situational awareness.
- Intelligence gathering: AI can be used to process and analyze data from a variety of sources, including cameras, sensors, and communication systems, to gather intelligence about enemy movements and activities.

Therefore, AI can help ground vehicles operate more efficiently, effectively, and safely on the battlefield, giving soldiers a strategic advantage in combat situations.

The rest of the article has the following structure: In Section 2, we first present an introduction to the AI inference along with recent trends for accelerating it in section 2.1. We discuss the traditional approach used for improving performance using parallel processing in section 2.2. In section 3, we introduce the concepts that form the backbone of the main contribution of this paper - namely the data pipelining using lock-free approach to AI inference. Section 3.1 introduces the concepts of lock-free algorithms. Section 3.2 introduces the pipelining approach and section 4 describes how these two ideas are combined to develop the novel AI inference engine. Section 5 describes the experimental setup used to test the inference engine and validate the performance benefits of this new approach.

2 INFERENCE IN ARTIFICIAL INTELLIGENCE

There are several components to artificial intelligence (Fig. 1). First, there is the training and design of the model based on the data to be analyzed. This activity is usually carried out by data scientists for a specific field of interest. Once the model is

designed, trained and validated to perform suitably on test data, it is deployed to the target computer for real-time execution. This is what is called inference. Inference consists of two parts, the trained model and the AI inference engine to execute the model. The focus of this research has been solely on the inference engine software algorithms.

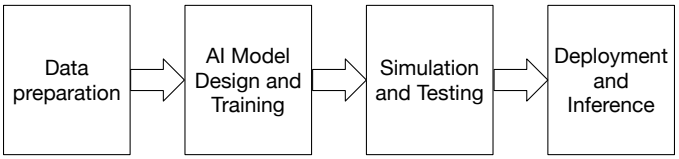

Figure 1: AI main components

2.1 Trends in Artificial Intelligence inference acceleration

The most common operation in AI inference by far is matrix multiplications. These operations are constantly repeated for each input data to the AI model. In recent years, there has been a substantial development in this area with both industry and academia progressing steadily in this field. While the current trend is to focus on hardware acceleration like Graphic Processing Units (GPU) [4] and Field-programmable gate array (FGPA) [5], these techniques are currently not broadly available to the Space industry due to radiation issues and excessive energy consumption for the former, and programming costs for the latter. The use of CPU for inference, however, has been also undergoing an important evolution taking advantage of modern Floating processing unit (FPU) connected to the CPU [6]. CPUs are widely used in Space due to large Space heritage and also ease of programming and use. Several AI inferences engines are available for CPU+FPU setups. The work presented here will show the results of extensive research in building a new AI inference that both reduces power consumption and also increases data throughput.

2.2 Parallel processing applied to Artificial Intelligence inference

Within the field of inference engines for CPU+FPU, the focus for performance optimisation has been on matrix multiplication parallelisation [7], [8]. This process consists in splitting the operations required for a matrix multiplication into smaller to be executed by several threads in parallel. Fig. 2 shows an example of this type of process, where rows from the left-hand matrix and columns from the right-hand matrix are individual operations to be executed by different threads.

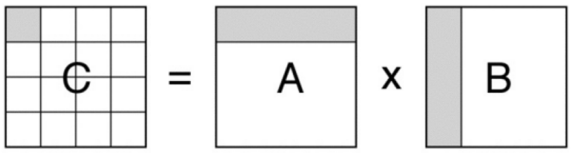

Figure 2: Parallel matrix multiplication

3 LOCK-FREE APPROACH TO AI INFERENCE

The work presented here is a based on a two dimensional threading model that combines two types of threads: lock-free pipelines and traditional parallelization techniques. The rest of this section explains both threading approaches and how they are combined to produce a novel high performance AI inference engine.

3.1 Lock-free parallel data processing for embedded systems

The work presented here is based on the concept of lock-free ring buffers [9]. Lock-free ring buffers are high performance concurrency framework based on CAS (Compare And Swap). They have drawn huge interest from industry due to its efficiency. For example, companies like LMAX have developed software using these techniques[10] that can, for instance, handle up to six million orders per second [11].

Lock-free programming in general, and ring buffers in particular, are traditionally written in Java. The reason for this is the presence of the Java Garbage Collector, which makes the very complex development of lock-free programming much more accessible to developers than it is in other languages. Moreover, the popularity of books like "Java Concurrency in Practice" [12], made a big impact in the Java development community. With the arrival of C++11 and smart pointers [13], it was possible to partially port ring buffers to C++, given the resemblance to Java Garbage Collector programming style.

3.1.1 Lock-free event loop

Based on this pattern and combining it with the new smart pointers in C++11 and wrapping it within a publisher subscriber pattern [14], we developed a simple application public interface (API), miniaturizing the almighty LMAX disruptor for embedded systems as presented in Fig. 3.

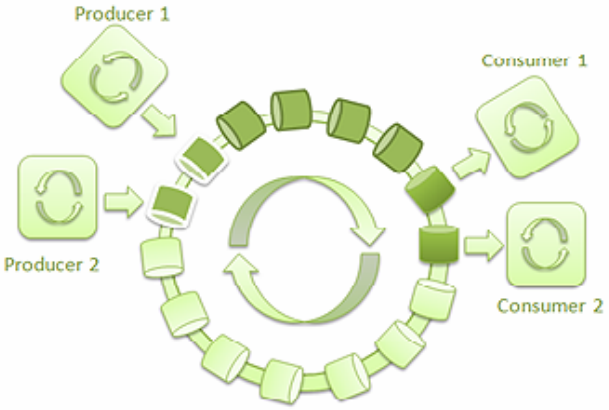

Figure 3: Generic ring buffer.

We limited the offered functionality of the disruptor to two main approaches. The first is the sensor multiplexer as in Fig. 4, which is a single producer, multiple consumer solution. Mainly used for vision navigation robotics and drone applications, this pattern is out of the scope of this article. The second, which we developed, is an eventloop (Fig. 5) loosely based on this development for financial systems. The eventloop presented here is lock-free, high performance and with a degree of determinism that has not been seen before in embedded systems [15].

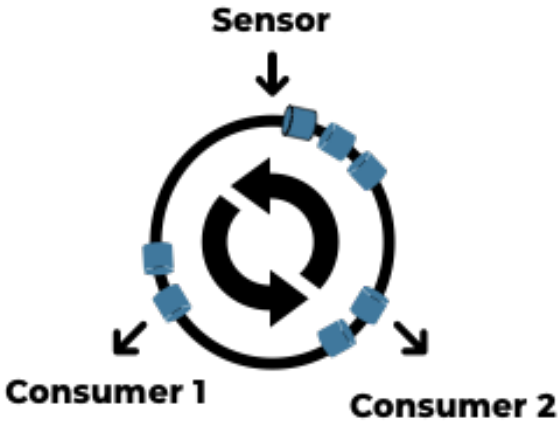

Figure 4: Sensor multiplexer - Special case of the generic ring buffer with only 1 producer and multiple consumers.

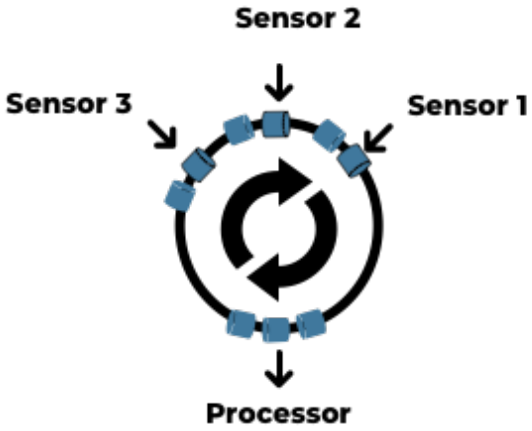

Figure 5: Lock-free eventloop - Special case of the generic ring buffer with multiple producers and single consumer.

The eventloop is a multiple producer, single consumer ring buffer which allows a fine degree of

control over memory usage. The lock-free design provides a low latency solution to process huge amounts of data. Applications like AI inference deal with large amounts of data and this lock-free design can be very powerful in improving the performance of inference engines.

3.2 Data pipelining

The theoretical advantage of the traditional parallelisation approach is its minimal latency [7]. However, there is an emerging alternative approach to parallelisation, which is based in the concept of pipelines [16]. This approach works in a similar manner to an assembly line, where each part of this line corresponds to a complete matrix multiplication. Fig. 6 shows this approach as applied to matrix multiplication as compared to the traditional parallelisation. The pipelining approach is particularly well suited for AI deep neural networks (DNN), since DNNs feature multiple ”layers” of computationally intensive calculations. The main advantage that makes pipeline a reliable approach to data processing in resources constraint environment, like Space on-board computers, is its higher throughput: pipelining can enable a substantial increase in throughput with respect to traditional parallelisation [17].

In the context of AI inference, pipelining consists of identifying separate groups of layers of the DNNs and treating them as separate ”blocks” of a pipeline. Data being processed by one ”block” need not be related to the data processed on another ”block”. When one block has finished processing its data, it sends it to the next block and is ready to accept new data for processing. Decoupling the various blocks allows us to then investigate and optimize each block separately based on the computational needs of that block.

4 THE NOVEL THREADING MODEL

Combining the concept of pipelining above with lock-free algorithms [18], the authors have developed a new pipelining approach that can process data at 2 to 8 times increased data rate, while at the same time reduce power consumption up to 75%. This new pipelining algorithm consists of three main elements:

- Use of lock-free ring-buffers to connect the matrix multiplication operations.
- Use of FPU vectorisation to accelerate the matrix multiplications
- One ring-buffer per thread, meaning that each matrix multiplication happens in one thread.

This novel approach can be seen in Fig. 7. Careful configuration of the eventloops allows allocating different memory and CPU resources to each layer of the DNN, which can help optimize the performance of the AI inference for latency, throughput or CPU utilization.

It must be noted that the underlying calculations for each layer in the DNN are not changed. As the only changes made are to the data transfer between layers, the inference accuracy of the original DNN is maintained. The accuracy of the DNN models was tested by comparing the output of Klepsydra AI against output obtained from using other inference engines (such as Tensorflow lite [19]) for the same input data.

4.1 Threading model optimisation

DNNs used for different applications can have widely differing architectures and the performance can vary based on number and type of layers used in the DNN. Additionally, the optimization constraints for performance may change based on the type of application for which the DNN is being used. For example, DNNs used for navigation may have hard constraints on latency and throughput. On the other hand, reducing CPU usage may be more important for non-critical applications like filtering low quality images captured by a camera before storage.

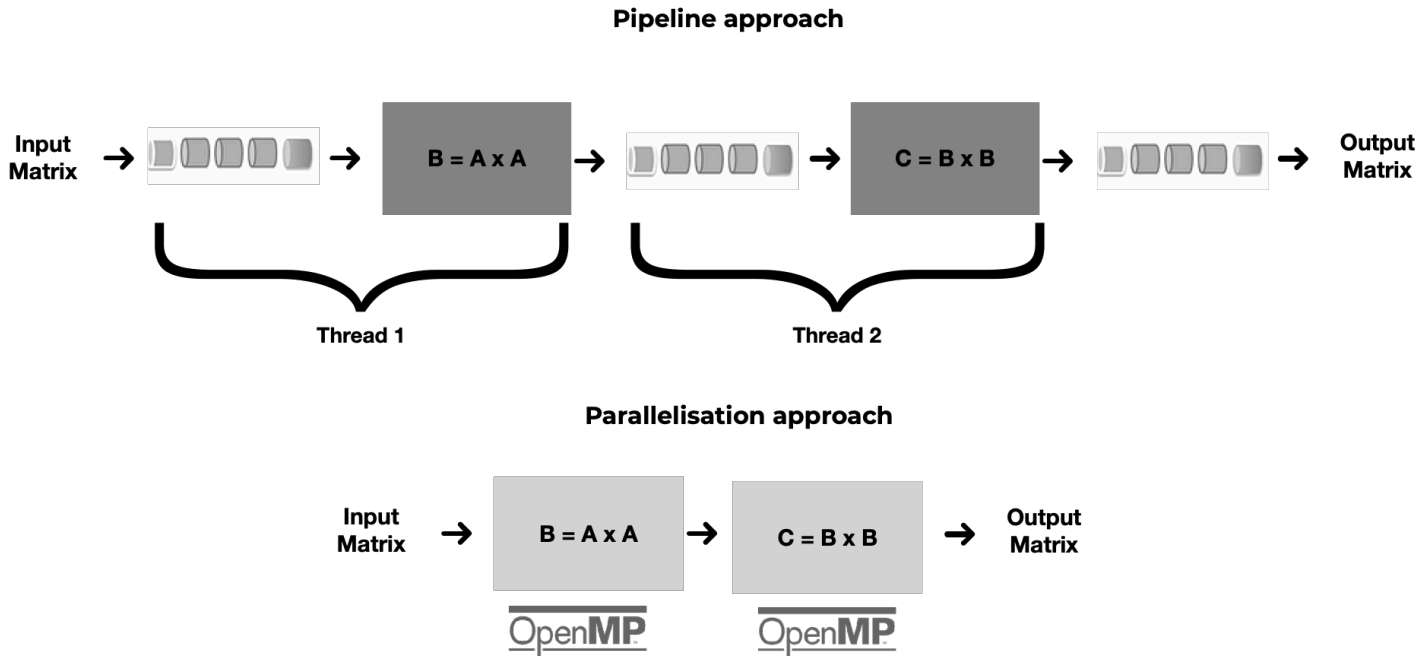

Figure 6: Pipelining vs parallelisation: Each operation in the pipeline only consumes part of the available resources. In parallelisation, each operation is parallelised to optimally use all available resources

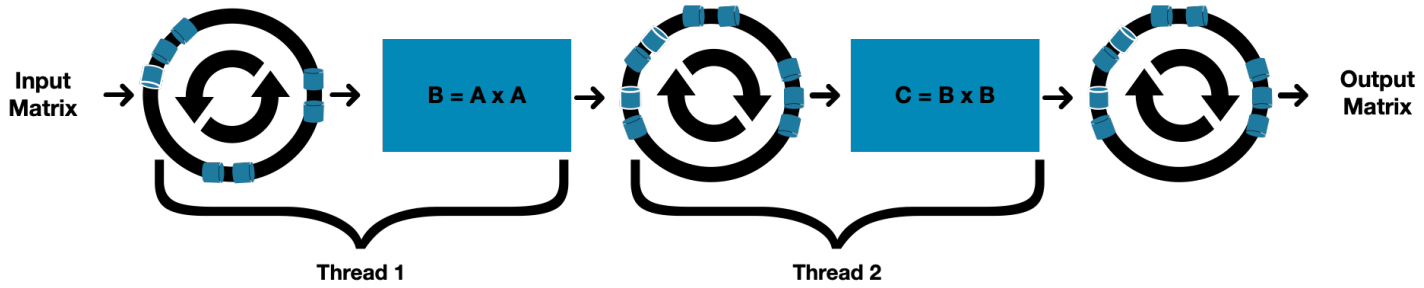

Figure 7: Novel proposed pipelining approach combining the pipelining with lock-free ring-buffers. Each operation runs in one thread, and the operations themselves use vectorised multiplications

Configuring eventloops for the application then becomes an important step of this threading model. For very deep DNNs, we may be able to group multiple resource-light layers to share the ring-buffer to avoid creating too many threads. The number of CPU cores available for FPU vectorization and the number of cores available for eventloop threads are two parameters that can affect the performance of AI inference.

The ring buffer can use an object pool which also affects the memory usage of the process. Additionally this may affect the maximum throughput of the system. Fewer eventloops allow us to dedicate more resources to parallelization operations to reduce latency at the cost of higher CPU usage. On the other hand, more eventloops allow better pipelining improving throughput.

We have developed an autotuning tool which tests the target DNN with different eventloop configurations and finds the best configuration for each optimization constraint - low latency, high throughput, low CPU and low memory requirements.

The autotuning tool has a web interface (see Fig. 8), allowing the user to select an AI model to test. The tool allows the user to select a memory pool size, the number of cores to be reserved for pipelining and the number of threads for parallelization operations. The user is expected to provide a range of frequencies at which new data will be input to the model, to evaluate the performance characteristics of the AI model.

The autotuning tool generates all possible valid combinations of configurations and tests these at the various input frequencies on the target device. In the example shown in Fig. 8, we have up to 2 cores for pipelining, and 2 threads for parallelization - giving us 4 possible combinations for each input frequency. Additionally, we can test with and without an object pool, giving us 4 additional combinations. When the user provides the range of frequencies at which new input data is fed to the DNN, the autotuning tool will test these 8 configurations for each of the frequencies.

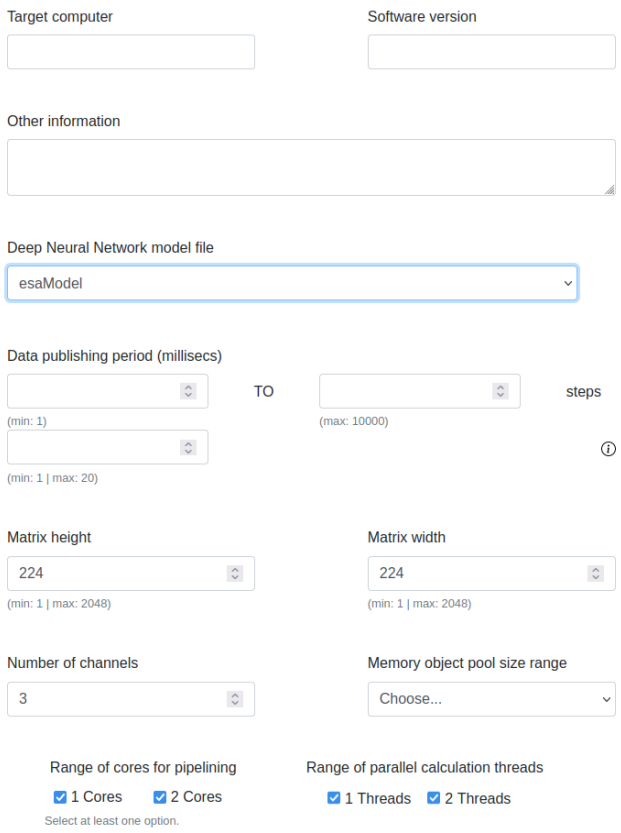

Figure 8: Autotuning tool configuration

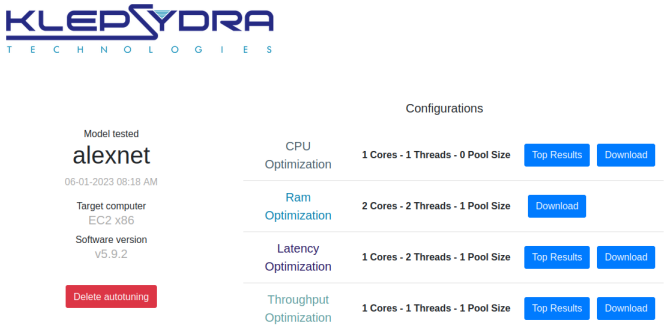

Figure 9: Best configurations for a sample Alexnet on 2-core Intel machine

The tool measures CPU usage, latency timings, RAM usage and finds the best configuration for each performance constraint. A sample result of best configurations for different criteria for an Alexnet is

shown in Fig. 9. The performance of the inference engine with each configuration is shown in Fig. 10.

These results of the autotuning tool allow us to compare the performance of an AI model under different criteria and choose the best threading model according to the final desired application.

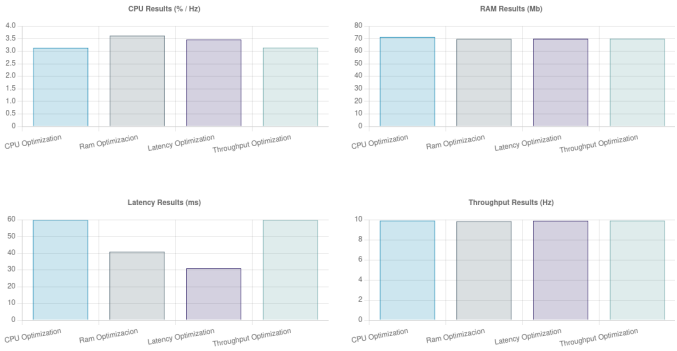

Figure 10: Sample performance for Alexnet on 2-core Intel machine for optimized configurations

5 THE EXPERIMENTAL SETUP

5.1 Overview

The performance results presented here were obtained during the European Space Agency (ESA) contract Klepsydra AI Technology Evaluation For Space Use (KATESU) [20].

The activity was aimed at porting the Klepsydra technology on a hardware target representative of on-board spacecraft, demonstrate it and evaluate its performance. Specifically, the objective of this activity was to port Klepsydra AI software into the specified on-board computer and specified operating system. Moreover, the software was tested and validated using first, standard models like Alexnet and MobileNet, and then followed by validation on AI models provided by ESA (namely CME [21] and U-Net[22] for Cloud detection). Validation consisted of comparing prediction obtained using Klepsydra AI software on given models for test data with the predictions obtained using existing market solutions. Finally, a performance benchmark and analysis was carried out testing ESA's DNNs in the above-mentioned setup and the results were compared with existing market solutions like TensorFlow Lite (TFL).

5.2 The technical setup

The presented solution was tested in the following setups:

5.2.1 Processors

- QorIQ® Layerscape LS1046A Multicore Processor. OS: Yocto Linux (Jethro). The reason for selecting this processor is that there exists a Space grade Radiation Tolerant version that is offered by Teledyne e2v[23].
- ZedBoard Zynq-7000 ARM/FPGA SoC Development Board; OS: Petalinux 2021.2

Docker containerisation was used in both processors. The setup elements are shown in Fig. 11.

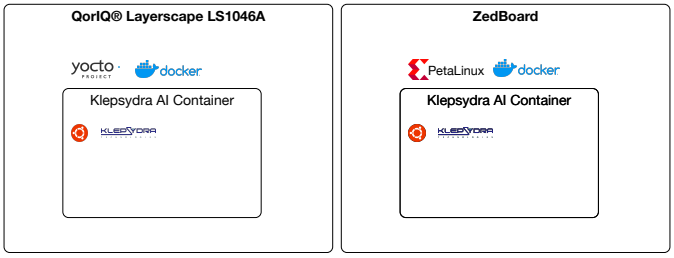

Figure 11: Processor setup for inference on LS1046A and Zedboard

5.2.2 AI Models

The Klepsydra AI framework, used for this study, is compatible with a variety of architectures including AI models for object detection and semantic segmentation. Performance benchmarks are presented in this article for the following models.

- Coronal Mass Ejections (CME) detection model[21] provided by ESA[24].
- U-Net for cloud detection [22]

The model for CME detection is a sequential model with convolutional networks, similar to Alexnet, whereas the U-Net has a graph architecture with skip connections. The CME detection model predicts a single floating point value showing whether there are Coronal Mass Ejections based on image data. The U-Net performs segmentation on image data and predicts an image highlighting zones covered by clouds. We chose these two models as they have different architectures and hence, different computational needs, and this allows us to highlight the different use cases for which the inference engine can be used.

5.3 DNN deployment steps

Since Klepsydra is fully compatible with the current 'de facto' standards for ML model, the deployment process consists of the following standard steps:

- Training of the model
- Quantization (if required)
- Final deployment in ONNX format [25]

For this activity, we obtained pretrained models in floating point and 8-bit quantized format. The Alexnet and Mobilenet models were run on synthetic data, while the CME and U-Net were run on real test data. The CME model was tested on test data provided by ESA [24], while the U-Net was tested on the data from the Cloud 95 dataset [26]. The LS1046 was used to test the full and quantized models both, whereas the Zedboard was used to test only the quantized models.

5.4 Performance results

5.4.1 LS1046

In order to simulate usage in real time, the models were evaluated by providing new input data at regular intervals. CPU usage was measured as normalized to the frequency of new input data, to account for differences in model complexities and throughput.

Performance results on the LS1046 for the CME model show a substantial increase in performance with a 4-fold increase in throughput, 30% CPU reduction and 66% reduction in latency with respect to TensorFlow Lite inference tool[19]. The quantized version of the CME model also showed a 4-fold increase in throughput, 25% CPU reduction and 4-fold reduction in latency. These results are summarized in Fig. 12 and Fig. 13.

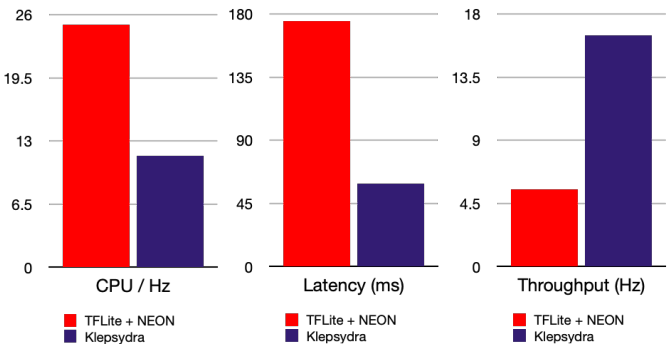

Figure 12: LS1046 Results, CME model

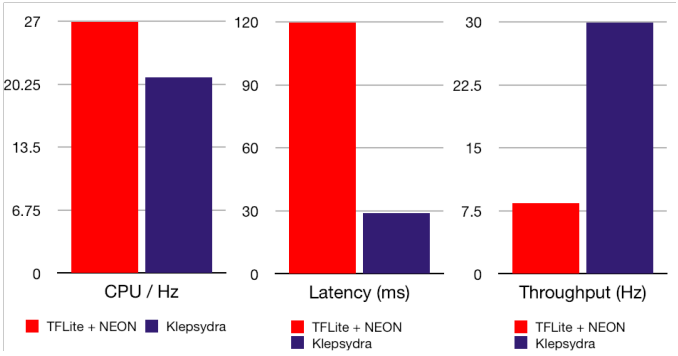

Figure 13: LS1046 Results, CME Quantized model model

For the U-Net model, we obtained 3-fold increase in throughput, 20% CPU reduction and 50% reduction in latency with respect to TensorFlow Lite. The results are shown in Fig. 14.

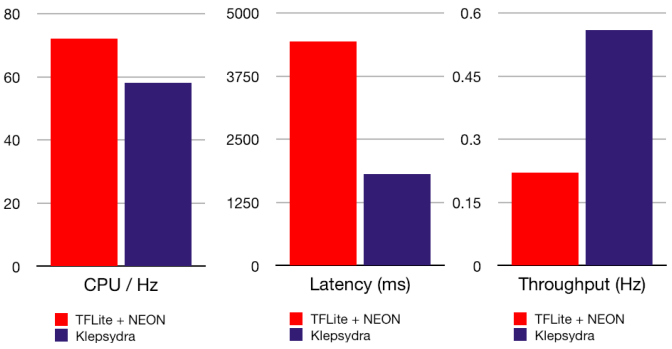

Figure 14: LS1046 Results, U-Net model

5.4.2 Zedboard

Only the quantized model was tested on the Zedboard. In case of the CME model, we obtained a 3-fold increase in throughput, 40% CPU reduction and 50% reduction in latency. The results are shown in Fig. 15.

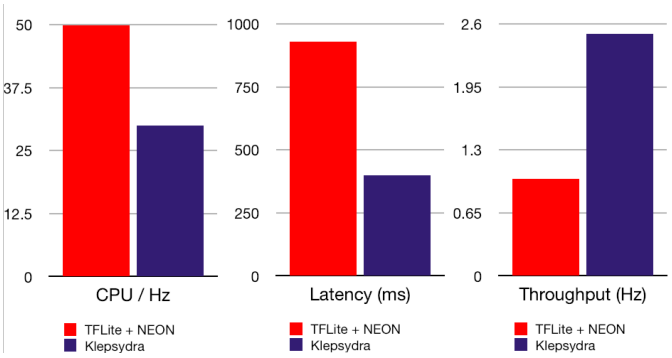

Figure 15: Zedboard Results, CME Quantized model model

6 CONCLUSIONS AND FUTURE WORK

Lock-free programming techniques, together with pipelining can bring three main benefits to on-board processing: Faster data processing, reduced power consumption on-board, and determinism. The benefits for ground vehicle system are clear for those areas needing large data processing: Autonomous navigation and intelligence gathering. The approach for AI inference described in this papers permits using deep neural networks for on-board processing without incurring significant resource costs, and without the necessity of upgrading rugged computer systems to expensive hardware.

The main areas of future research work for the short term are: adding FreeRTOS support, NVIDIA GPU Support, and extending this approach to FPGA. Moreover, for the mid-term plan, we plan to add support for Reinforcement Learning, support for running inference on RTEMS v5 and also perform Space pre-qualification of the inference engine.

7. REFERENCES

[1] P. Sharma, H. Verma, V. Negi, A. Sharma, S. Banarwal, and G. Verma, "Evolutionary trends in embedded system design," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, March 2016, pp. 2059–2062.

[2] F. Samie, L. Bauer, and J. Henkel, "Iot technologies for embedded computing: A survey," in *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Oct 2016, pp. 1–10.

[3] A. Banerjee, A. Mondal, A. Sarkar, and S. Biswas, "Real-time embedded systems analysis — from theory to practice," in *2015 19th International Symposium on VLSI Design and Test*, June 2015, pp. 1–2.

[4] T. Baji, "Evolution of the gpu device widely used in ai and massive parallel processing," in *2018 IEEE 2nd Electron Devices Technology and Manufacturing Conference (EDTM)*, 2018, pp. 7–9.

[5] A. Shawahna, S. M. Sait, and A. El-Maleh, "Fpga-based accelerators of deep learning networks for learning and classification: A review," *IEEE Access*, vol. 7, pp. 7823–7859, 2019.

[6] H. Lan, J. Meng, C. Hundt, B. Schmidt, M. Deng, X. Wang, W. Liu, Y. Qiao, and S. Feng, "Feathercnn: Fast inference computation with tensorgemm on arm architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 580–594, 2020.

[7] L. Dagum and R. Menon, "Openmp: an industry standard api for shared-memory programming," *IEEE Computational Science and Engineering*, vol. 5, no. 1, pp. 46–55, 1998.

[8] S. Shukla, B. Fleischer, M. Ziegler, J. Silberman, J. Oh, V. Srinivasan, J. Choi, S. Mueller, A. Agrawal, T. Babinsky, N. Cao, C.-Y. Chen, P. Chuang, T. Fox, G. Gristede, M. Guillorn, H. Haynie, M. Klaiber, D. Lee, S.-H. Lo, G. Maier, M. Scheuermann, S. Venkataramani, C. Vezyrtzis, N. Wang, F. Yee, C. Zhou, P.-F. Lu, B. Curran, L. Chang, and K. Gopalakrishnan, "A scalable multi-teraops core for ai training and inference," *IEEE Solid-State Circuits Letters*, vol. 1, no. 12, pp. 217–220, 2018.

[9] P. P. C. Lee, T. Bu, and G. Chandranmenon, "A lock-free, cache-efficient multi-core synchronization mechanism for line-rate network traffic monitoring," in *2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, 2010, pp. 1–12.

[10] M. Baker and M. Thompson, "Lmax disruptor," 2011. [Online]. Available: \url{http://github.com/LMAX-Exchange/disruptor}

[11] Y. Fang, H. Zhu, F. Zeyda, and Y. Fei, "Modeling and analysis of the disruptor framework in csp," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, 2018, pp. 803–809.

[12] T. Peierls, B. Goetz, J. Bloch, J. Bowbeer, D. Lea, and D. Holmes, *Java Concurrency in Practice*. Addison-Wesley Professional, 2005.

[13] R. Kaiser, *C++11 Smart Pointer: shared ptr, unique ptr und weak ptr*. https://www.rkaiser.de/, 01 2019, pp. 781–799.

[14] P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, pp. 114–131, 06 2003.

[15] T. A Henzinger, "Two challenges in embedded systems design: Predictability and robustness," *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 366, pp. 3727–36, 11 2008.

[16] G. Biggs, N. Ando, and T. Kotoku, "Rapid data processing pipeline development using openrtm-aist," in *2011 IEEE/SICE International Symposium on System Integration (SII)*, 2011, pp. 312–317.

[17] K. Sravani and R. Rao, "High throughput and high capacity asynchronous pipeline using hybrid logic," in *2017 International Conference on Innovations in Electronics, Signal Processing and Communication (IESC)*, 2017, pp. 11–15.

[18] P. Ghiglino and M. Harshe, "A deterministic and high performance parallel data processing approach to increase guidance navigation and control robustness." in *2020 IAF SPACE SYSTEMS SYMPOSIUM (IAC)*, 2020.

[19] Google, "Tensor flow lite," https://www.tensorflow.org/lite, 2015.

[20] P. Ghiglino, "Katesu - klepsydra ai technology evaluation for space use," https://indico.esa.int/event/404, 2022.

[21] R. Dengel, "Methods and deployment of machine learning on several embedded systems," in *15th ESA Workshop on Avionics, Data, Control and Software Systems (ADCSS2021)*, 2021.

[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015. [Online]. Available: https://arxiv.org/abs/1505.04597

[23] T. e2v, "ls1046 space," https://semiconductors.teledyneimaging.com/en/products/processors/ls1046-space, 2019.

[24] D. Valsesia, A. Grippi, E. Magli, R. Susino, D. Telloni, G. Nicolini, M. Casti, A. F. Mulone, and R. Messineo, "Detection of solar coronal mass ejections from raw images with deep convolutional neural networks," in *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, 2020, pp. 2272–2275.

[25] J. Bai, F. Lu, K. Zhang *et al.*, "Onnx: Open neural network exchange," https://github.com/onnx/onnx, 2019.

[26] S. Mohajerani and P. Saeedi, "Cloud-Net+: A Cloud Segmentation CNN for Landsat 8 Remote Sensing Imagery Optimized with Filtered Jaccard Loss Function," vol. 2001.08768, 2020.