UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

Final Project in *Real-Time System and Industrial Applications*

Performance and Interference Analysis of AI Workloads on the BeagleBone AI-64 Board

Anno Accademico 2024/2025

Candidates
Vincenzo Luigi Bruno matr. M63001670
Salvatore Cangiano matr. M63001647

Abstract

This project presents a detailed analysis of AI model inference performance and real-time task latency on an edge device, the BeagleBone AI-64, running a Debian 11 environment with Linux Kernel 5.10.168. The primary objective is to quantify the impact of various system stressors on the throughput and efficiency of AI workloads, accelerated via the TI Deep Learning (TID) delegate on C7x/MMA coprocessors, and on the predictability of real-time tasks measured with cyclictest.

The experimental methodology involved measuring inference times for six pre-trained models (MobileNet v1/v2, SSD MobileNet v1/v2 and Resnet18/Resnet50) under baseline conditions and while subjected to seven distinct stress-ng stressors (cpu, vm, memcpy, interrupt, open, fork, udp). Each model-stressor combination was replicated 30 times to ensure statistical significance, with careful management of the Linux page cache between repetitions. In conclusion, this study provides a quantification of the impact of various forms of system interference on AI and real-time performance in an edge context, highlighting the importance of managing main processor and kernel resources to ensure predictability and efficiency in critical applications.

Contents

Chapter 1

Experimental Setup

1.1 Hardware Platform: BeagleBone AI-64

The platform used for this study is the **BeagleBone AI-64**, a single-board computer designed for high-performance edge AI application. Its architecture is centered around the **Texas Instruments *TDA4VM* SoC**, which features a heterogeneous collection of processing units.

The key hardware components relevant to this study are:

- **Main Processor Unit (MPU)**: A dual-core 64-bit **Arm Cortex -A72** microprocessor subsystem operating at 2.0 GHz, which runs the main Linux OS and user-space applications.
- **AI Accelerators**: The SoC integrates a **C71x Digital Signal Processor (DSP)** and a **Matrix Multiplication Accelerator (MMA)**. These co-processors are designed to offload and accelerate deep learning computations.
- **Memory**: The board is equipped with **4GB of LPDDR4 RAM** and **16GB of onboard eMMC flash storage**. *It is important to note that approximately half of the system RAM is reserved for use by the hardware accelerators.*

  To ensure stable and consistent performance during testing, the board was powered by a dedicated 5V, 3A DC power supply connected via the barrel jack, as recommended by the official documentation.

1.2 Software Environment

The software stack was configured to provide a stable and repeatable environment for performance benchmarking

- **OS**:The board was flashed with the official **`BBAI64 11.8 2023-10-07 10GB eMMC TI EDGEAI Xfce Flasher`** image. This provides a **Debian 11 (Bullseye)** based environment running the **Linux Kernel version 5.10.168-ti-arm64-r11**.
- **AI Stack**: The pre-installed **TI Edge AI SDK**, located in `/opt/edge_ai_apps`, served as the foundation for running AI models. Inference is accelerated by offloading model subgraphs to the C7x/MMA hardware via the **TI Deep Learning (TID) delegate** ((libtidl_tfl_delegate.so)
- **Benchmarking Tools**: `stress-ng` and `cyclictest`.

1.3 AI Workloads and Stressors

The following four pre-trained models were selected to cover different architectures and tasks (image classification and object detection):

1. **MobileNet v1** (trained on ImageNet)
2. **MobileNet v2** (trained on ImageNet)
3. **SSD MobileNet v1** (trained on COCO)
4. **SSD MobileNet v2** (trained on COCO)
5. **ResNet18** (trained on ImageNet)
6. **ResNet50** (trained on ImageNet)

The following seven stressors from the `stress-ng` suite were used to create different types of system interference, in addition to a "no-stress" baseline condition:

1. **cpu**: Stresses the floating-point and integer units of the CPU cores.

2. **vm**: Stresses the virtual memory subsystem by allocating and manipulating memory.

3. **memcpy**: Stresses the memory bus by performing large memory-to-memory copy operations.

4. **interrupt**: Generates a high rate of hardware interrupts.

5. **open**: Stresses the filesystem by repeatedly opening and closing files.

6. **fork**: Stresses the process scheduler by rapidly creating and destroying new processes.

7. **udp**: Stresses the kernel's network stack with local UDP traffic.

Chapter 2

Test Methodology

2.1 Performance Metrics (KPIs)

To quantitatively evaluate performance, the following KPIs were selected:

- **For AI Workloads**: The primary metric is the **Inference Time**, measured in millisecons (ms). This represents the time required for the model to process a single input frame and is a direct indicator of the AI accelerator's throughput and efficiency. Data was collected using the logging tools provided by the TI Edge AI SDK, which record performance statistics from the C7x/MMA hardware accelerators.

- **For Real-Time Tasks**: The primary metric is the **Scheduling Latency**, measured in microseconds by the `cyclictest` tool. Specific attention was given to the Maximum Latency, as this value represents the worst-case delay experienced by the task and is the most critical parameter for systems with hard real-time constraints.

2.2 Impact of Stressors on AI Performance

The experiment began by first establishing a performance **baseline** for each of the four selected AI models. This was achieved by executing them on an idle system to record their inference times under optimal conditions.

Subsequently, the test were repeated for each model while a specific system **stressor** was running in parallel to measure the performance degradation under various types of resource contention. To ensure statistical significance and indipendence between measurements, **30 repetitions** were performed for each combination of model and stress condition.

The entire process was automated via a Bash script to maintain consistency across tests. For each run, the application processed a fixed set of **90 input images**.

Crucially, to mitigate caching effects and ensure that each run started from a comparable system state, the Linux page cache was dropped and a brief sleep interval was enforced between each repetition.

The resulting performance logs were then saved for subsequent analysis.

This procedure allowed us to quantify how different sources of system interference affect the performance and predictability of AI tasks running on the dedicated hardware accelerators.

2.3 Impact of AI workload on Real-Time Task Performance

In this case study, the focus shifts to understanding how an AI workload can affect a real-time task, simulated with cyclictest.

Following a similar methodology of the previous paragraph, the baseline performance of cyclictest was initially measured without any interference. Subsequently, for each of the already analyzed models, an AI workload was executed in the background during the cyclictest execution.

A key objective of this experiment was to determine the influence of an AI workload on real-time tasks and to investigate if any such impact could be limited. To address this, two configurations for the measurement environment were established:

- **Non-Isolated CPU**: In this configuration, cyclictest was executed within the standard system environment, identical to that used in previous sections.
- **Isolated CPU**: For this setup, the environment was configured to provide a low-latency execution core. **Core 1 was isolated** from the general kernel scheduler

by appending the boot parameters `isolcpus=1` and `rcu_nocbs=1` to the bootloader configuration file located at `/boot/firmware/extlinux/extlinux.conf`. The `isolcpus` parameter removes the core from general task scheduling, while `rcu_nocbs` offloads RCU (Read-Copy-Update) callback processing to non-isolated cores, minimizing a key source of non-deterministic latency.

Note on Isolation Configuration

It should be noted that the `nohz_full=1` parameter was omitted, as analysis confirmed the running kernel was not compiled with the requisite `CONFIG_NO_HZ_FULL` option. Furthermore, an analysis of interrupt affinity showed that the per-CPU `arch_timer` interrupt (IRQ 11) is non-migratable and remains on the isolated core by design. The kernel locks the affinity of this fundamental timer to its designated core to ensure system stability.

To complete the setup, the `cyclictest` process was pinned exclusively to the isolated core using the `taskset -c 1` utility, ensuring that all measurements for this scenario were taken within the prepared low-latency environment.

Chapter 3

Results and Analysis

To present the gathered KPIs, boxplots were utilized. Their visual strength lies in their ability to directly convey key distributional characteristics.

3.1 Analysis of AI Inference Performance

Measurements of inference time capture the entire duration of an inference cycle, from the time the input is prepared until the result is available. This includes several critical steps:

1. **Input Data Transfer**: The copying of input data from main processor memory to the dedicated memory of the artificial intelligence accelerator.
2. **Inference Execution**: The actual processing of the model on the dedicated hardware accelerator.
3. **Output Data Transfer**: The reading of the inference results from the accelerator memory and making them available to the main processor.
4. **Runtime Overhead**: The internal operations handled by the runtime software to coordinate the accelerator and manage the flow of data and commands.
5. **Synchronization**: The overhead associated with internal synchronization mechanisms (e.g., locks) that, while minimal for a single inference, contribute to overall latency.

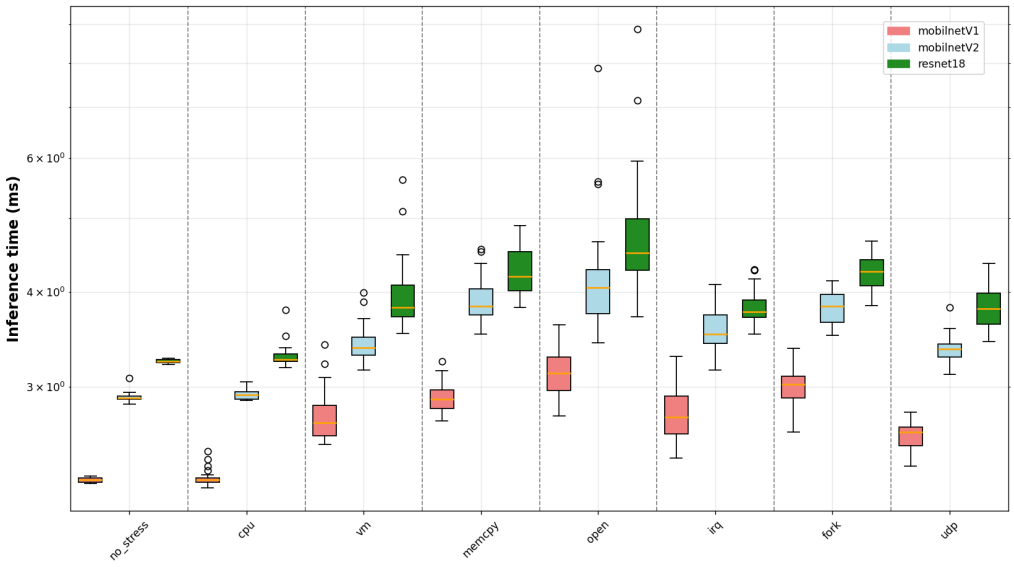

Figure 3.1: MobilenetV1, MobilenetV2 and Resnet18 Inference times

This graph vividly illustrates the impact of various stressors on inference times. For MobileNetV1, MobileNetV2 and Resnet18 models, baseline inference times (under `no_stress` conditions) consistently fall within the millisecond range.

Stressors not only cause an observable increase in the median inference time—typically by 3 or 4 milliseconds compared to the `no_stress` case—but also significantly widen the spread of the data, as indicated by the increased interquartile range

- **Impact of `vm` and `memcpy` Stressors**: Even with dedicated memory for the AI accelerator, these stressors appear to induce a significant increase in inference times. It can be hypothesized that this is not due to a slowdown in computation on the accelerator itself, but rather to contention for bandwidth or availability of the main processor's memory. The input and output data transfer operations, which occur on the shared memory bus, could become a crucial bottleneck, thereby prolonging the total measured time.

- **Impact of `open`, `irq` , and `fork` Stressors**: These stressors seem to directly influence the performance of the main processor and the kernel subsystem. The observed increase in inference times in these scenarios could be attributed to:

  - **Main Processor Contention**: If the main processor is overloaded, its ability

to allocate processing cycles to the thread managing the inference pipeline might be reduced.

– **Kernel Call Latency**: Interactions between the runtime software and the AI accelerator's hardware driver involve frequent kernel calls. When the kernel is under pressure due to these stressors (interrupts, in particular, could be critical for hardware communication), the latency of these system calls might increase, directly impacting the overall inference time.

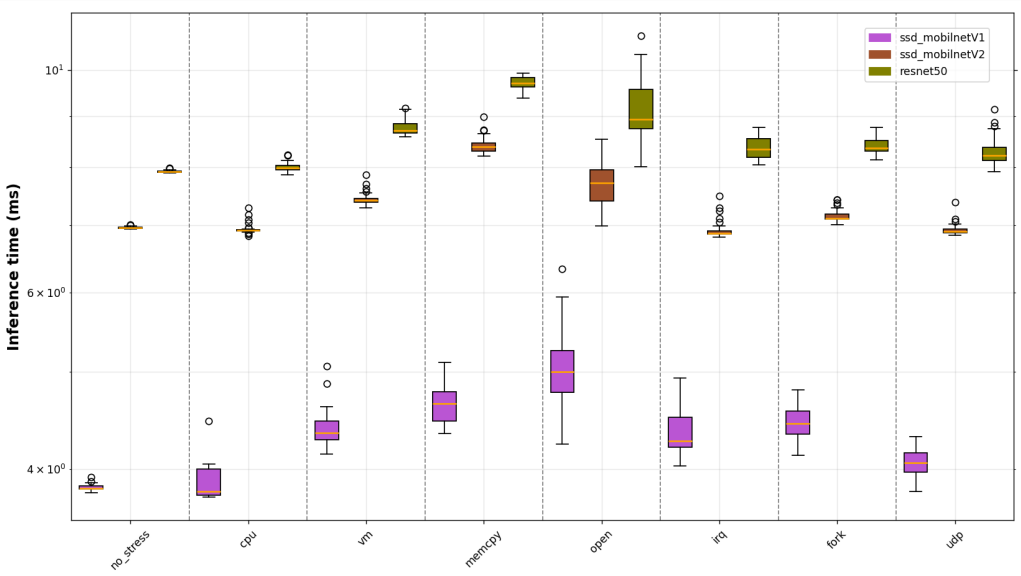

Figure 3.2: ssd_MobilenetV1, ssd_MobilenetV2 and Resnet50 Inference times

A similar behavior is observed in the SSD variants of the MobileNet models and for Resnet50. As expected, both SSD MobileNetV1 and SSD MobileNetV2 exhibit higher baseline `no_stress` inference times compared to their non-SSD counterparts, a consequence of their increased complexity. While SSD MobileNetV2 might appear to have a narrower interquartile range under stressors like `irq` and `fork`, the presence of numerous outliers in these conditions consistently supports our previous observations regarding increased variability under stress.

3.2 Analysis of Real-Time Task Latency

As the previous views, it has been plotted maximum latency values of cyclic test in a Isolated Enviroment and in a Non-Isolated Enviroment.

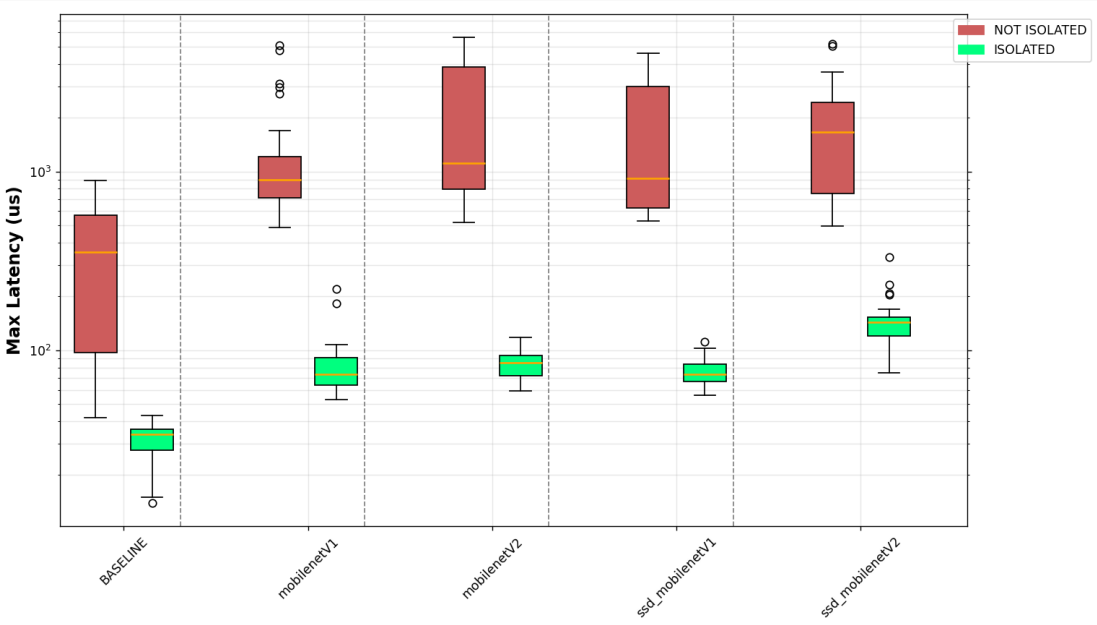

Figure 3.3: Cyclictest Boxplots

The results unequivocally demonstrate that even in the absence of AI workloads, the interquartile ranges in non-isolated CPU configurations are consistently larger compared to those observed in isolated environments. While it is generally true that the introduction of workloads increases latency, this increase is notably more contained and smaller in the isolated case.

It is important to note, however, that these promising results were achieved with a static configuration of bootloader variables, which may not represent the optimal setup for a real-world real-time task. This suggests a strong potential for achieving even superior performance with alternative configurations, such as those leveraging a Real-Time Operating System (RTOS) or Real-Time Virtualization technologies.

3.3 PREEMPT_RT Experiment

To further explore the real-time limits of the platform, a supplementary experiment was conducted using a kernel with the **PREEMPT_RT** patch. A version incompatibility was discovered: the provided TI Edge AI SDK for AI inference required Kernel `5.10`, while supported **PREEMPT_RT** patches for the board start from later versions (`6.1+`).

It was therefore decided to compile a newer kernel with the patch applied to validate its performance.

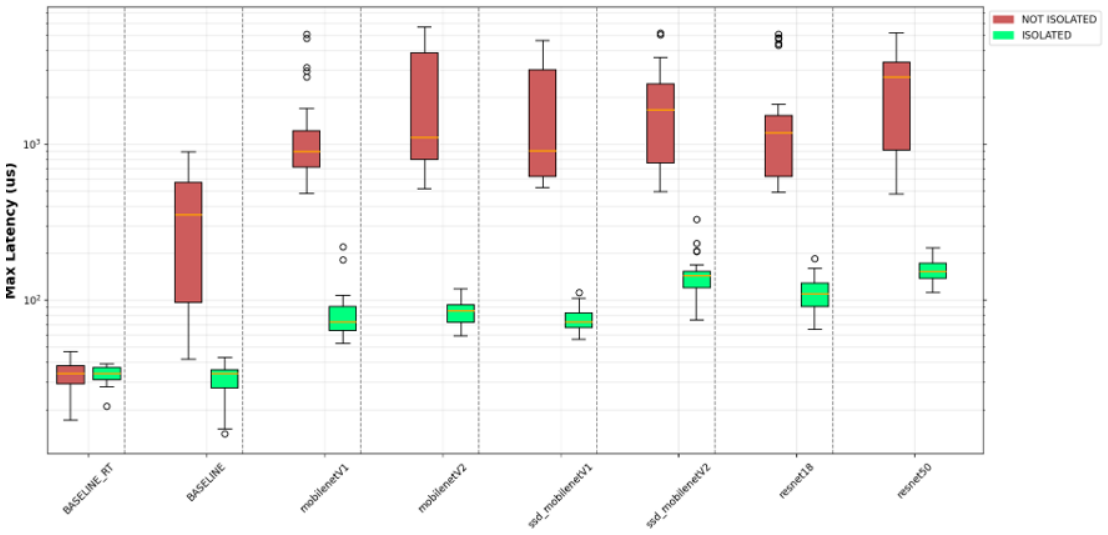

Figure 3.4: Cyclictest Boxplots

Although the incompatibility with the SDK prevented the execution of AI workloads, baseline measurements with `cyclictest` revealed a maximum latency and variability significantly lower than even the isolated CPU scenario, confirming the superiority of the **PREEMPT_RT** solution for applications with strict timing constraints.

Chapter 4

Conclusion

This study has quantified the bidirectional interference between artificial intelligence workloads and system-level tasks on a heterogeneous edge platform, the BeagleBone AI-64.

It was demonstrated how stress on system resources, particularly the I/O and memory, degrades the performance and predictability of AI models, even when they are executed on dedicated hardware accelerators. Likewise, the execution of these same AI workloads introduces significant interference that increases the maximum latency of real-time tasks, compromising their predictability.

Mitigation strategies proved to be essential. **Isolating a CPU core** using the `isolcpus` boot parameter was shown to be a practical and highly effective solution for protecting critical tasks from system and application interference. The supplementary experiment with the **PREEMPT_RT kernel**, while highlighting current software compatibility challenges, confirmed its technical superiority in providing a minimal and deterministic latency environment.