Measure energy consumption

INTRODUCTION:

Artificial Intelligence (AI) and Internet of Things (IoT) are two of the most prominent areas in technology research and innovation, and are predicted to grow in the future. Combining AI and IoT results in a paradigm called Artificial Intelligence of Things (AIoT) which comes with its own set of challenges and opportunities. Different areas related to AIoT in today’s society (e.g., healthcare and manufacturing) need specific solutions, optimizing for how much computation power and storage space is consumed and needed.

The goal of this study is to investigate the impact on performance (i.e., execution time, CPU usage, and memory consumption) and energy consumption of containerization strategies for AI-based computer-vision models on the Edge. To accomplish our goal, we run pre-trained models on physical devices, comparing the collected measures for three deployment strategies: the ONNX Runtime, Scailable's WebAssembly (WASM) engine, and Docker.

The main contributions of this study are: an empirical investigation on the performance and energy consumption of off-the-shelf AI deployment strategies, a discussion of the obtained results, and the replication package of the study containing all data, the source code for running the experiment, and the scripts for data analysis. The target audience of this study are AI/ML researchers and software engineers interested in the sustainability aspects of AI development and its potential impact on the environment.

BACKGROUND:

AI-based Computer vision on the Edge: In Computer vision, inference consists in an analysis (as a human) of visual data that leads to information on what the said data contains. It is considered a valuable area for research in Artificial Intelligence and in Computer Science in general, given the dynamics of the market . AI on the Edge comes with its challenges, opportunities and a novel set of use cases catered to environments with bandwidth and latency constraints. Computer vision includes tasks such as object detection, image identification and image classification. In healthcare, we encounter application providing cancerous moles detection in skin images or finding symptoms in x-ray and MRI scans. In agriculture, Edge AI could be employed for plant health monitoring. In manufacturing, Edge sensor devices can evaluate whether a component is defective.

ONNX Runtime: ONNX is an open ecosystem for interoperable AI models. Given its open-source nature, as well as its support and performance in the field of ML, the ONNX Runtime is considered a good candidate for this study. This containerization strategy has no additional layers apart from the inference engine, so we use it as the baseline containerization strategy in our experiment. With the arrival of Microsoft's ONNX Runtime, a high-performance inference engine is now available to make use of the the ONNX format. An .onnx file of a trained model contains the needed information to be able to switch between platforms and instantiate a full data processing pipeline (via the ONNX Runtime). To perform predictions on Edge devices, developers simply need the .onnx file and the input data.

WebAssembly: WebAssembly is a binary instruction format designed as a portable compilation target. While being more frequently employed in most modern browsers, there are a number of standalone runtimes that provide software containerization on any target CPU architecture. Scailable promises faster execution time and smaller memory footprint when separating training from prediction generation. We use Scailable to compile ONNX models into WASM binaries. The platform toolchain provides easy blackbox conversion of a fitted model from .onnx format to a dedicated, fully portable and highly optimized .wasm binary. Thus, the Scailable engine used in the experiment is essentially a small footprint, standalone pre-compiled runtime for an ARM64 CPU, based on wamr2. The engine supports both

.wasm and .aot files (equivalents of the .onnx files mentioned before), and we used the wamr compiler to convert .wasm files to their .aot version to ensure best possible performance. The motivation behind this choice is that Ahead-of-Time (AOT) compilation turns the WASM bytecode to a native module and decreases the amount of computation necessary during run time by performing a series of optimizations.

EXPERIMENT DESIGN

Goal :

The goal of the experiment is to analyse AI containerization strategies for the purpose of evaluating how they impact energy consumption and performance from the point of view of AIoT developers and researchers, in the context of computer-vision tasks in Artificial Intelligence on Edge.

Variables and statistical hypotheses:

- **Energy consumption**: considering µi as the median of the energy consumed by the deployment strategy I
- **Execution time:** being µi the median of the execution time of the deployment strategy i
- **CPU usage:** being µi the median of the CPU percentage utilised by the deployment strategy i

• **Memory usage:** being µi the median of the memory utilised by the deployment strategy i

For each dependent variable, the null hypothesis claims that there is no significant difference between the three containerization strategies, while the alternate hypothesis claims that for at least one pair of two treatments there is a significant difference. The hypotheses are not implying any direction on the studied phenomenon, so that no bias is introduced.

Data analysis

To analyze the outcome of the experiment, the following set of statistical methods are applied. We begin by performing preliminary observations on the measures using a series of plots to assess the distribution of data. Under the initial assumption that the collected data is not normally distributed we perform the Shapiro-Wilks test and visualize the data through Q-Q plots for each of the collected metrics. We can then decide if we further apply parametric or non-parametric tests to test the hypotheses.

EXPERIMENT EXECUTION

• 1: the main.py script will run the current iteration configuration <model, strategy>

and start by first communicating with the power monitor through hvpm.py;

• 2: a separate thread using hvpm.py starts periodic sampling using the HVPM;

• 3: the power monitor communicates directly with the device to both provide input voltage and record measures;

• 4: back in main.py, based on the strategy needed for this run a decision needs to be made on the script to be run next;

• 5: for docker, we separately start the container before running measure.py (5*); for all strategies, the measure.py script will first start the performance profilers on the device and log the raw data;

• 6: at this point all profilers are setup and we run the subject using as inputs the model file and the dataset already available on the device; to start the engine we use run.py for docker and onnxr (as they both use the same Python package) and the sclb-bin for wasm.

RESULTS

We firstly overview the raw measures, and then we present the results. The following abbreviations are used in the tables and figures: en – energy consumption, ex – execution time, rss – resident set size (memory), vsz – virtual set size (memory), pcpu – cpu measures collected via pdistat, tcpu – cpu measures collected via top.

THREATS TO VALIDITY

Internal Validity: The risk of how history might affect the results is reduced by performing the measurement all at once, in the same environment and using the same infrastructure, but in a randomized manner. To mitigate, we built scripts to automate the execution of the experiment. In this scenario, maturation comes as another possible threat, since the runs are performed repeatedly (and sequentially) on the same object. To mitigate this threat, we set an interval of 60 seconds between runs.

The selection of subjects (models) might constitute another potential threat as it was performed manually. Thus, the selected set of models could not be regarded as an accurate representative for the whole population. A first iteration in this process consisted in selecting a set of models from the ONNX Zoo, from each of the classes mentioned on the GitHub page, initially aiming to have two representatives from each class. However, we observed the lack of already trained models for some of the classes and what is more, most of the models were not suitable for conversion from .onnx to .wasm format .As a result to that, we ended up with a selection of 4 models from the Computer Vision (CV) category and the threat is reduced by aiming to diversify the

CV tasks they performed. It is worth mentioning that both prediction engines (ONNX Runtime and WebAssembly) perform worse on first input received compared to subsequent inputs, which is mainly because of a required warm up inference. We did not perform any special treatment for that initial inference to better represent real life deployments and, since we randomize input order for every run, we believe it does not introduce significant bias.

External Validity: The interaction of selection and treatment is one of the external threats concerning this study, also mentioned in the previous section. This threat can be mitigated in the future by repeating the experiment on more model classes, as they become available and allow conversion, or even making the training of the new models as part of a future experiment. In general, threats to external validity are reduced by providing a relevant and realistic measurement environment to minimize the effect of the interaction of setting and treatment. In our study we support the mitigation of this threat by performing the experiment on both a less (Raspberry Pi 3) and more (Raspberry Pi 4) highly performing device. On both devices we use a relatively recent and modern version of Ubuntu Server and latest versions for all the tools and packages employed in the instrumentation process. What is more, the generalization factor can be improved by replicating the experiment on other edge devices, some especially made for AI deployments such as the Jetson Nano.

Conclusion Validity: In our study, most of the p-values resulted from the applied statistical methods (such as Saphiro-Wilks and Kruskal-Wallis) are significantly lower that the chosen significance level of $\alpha = 0.05$. Additionally, we check the assumptions of the Kruskal-Wallis test (in Section 5) to ensure that those are not violated and to further support the correctness of our conclusions. Including all the data collected during the statistical analysis phase, especially the outliers, supports our claim of correctness by not fishing for a specific result. What is more, the replication package associated with this experiment allows the reproduction and validation of the outcomes presented in this paper.

CONCLUSION

This is aims to be an additional stepping stone in the field of AIoT, with a focus on the impact of three containerization strategies wrt energy consumption and performance. Scailable’s WASM engine (although new to the field) proved to be a relevant contender to ONNX and Docker. Based on the results of our experiment, for Edge AI applications with strict disk space and RAM memory requirements developers should prefer WebAssembly for deployment over ONNX or Docker; the ONNX strategy should be the preferred choice when energy consumption and execution time are a concern.