Embedded Benchmarks: An Overview

Motivation

Benchmarking is a very important topic in computer science. Benchmarks are used to measure the relative performance of systems, programs, or other operations. This performance is measured through a variety of standardized tests and trials that measure, such as time taken to complete a task, or the power consumption of an operation.

Benchmarks act as a figure of merit that help to determine the superiority of a particular approach, which can often dictate the path of future research. A good example of this is AlexNet, which debuted at the ImageNet Large Scale Visual Recognition Challenge in 2012. AlexNet was the prototypical deep neural network, the world's first introduction to deep learning, and it's statistical power was demonstrated clearly through benchmarking.

The ImageNet challenge used classification accuracy on the ImageNet dataset as the performance benchmark to assess the model submissions of contestants. AlexNet achieved an error rate of 15.3%. This error rate would be quite poor today, but at the time, the second best model had an error rate of 26.1% - this is 10.8% higher! Given that ImageNet consists of 1 million images, this means that it correctly classified approximately 100,000 more images than its closest competitor.

But What Constitutes A Benchmark?

A benchmark means different things to different communities. In the ML community, a benchmark refers to a dataset with some associated metrics attached to it (e.g., accuracy, model size) to measure model performance. To the hardware community, a benchmark is a set workload to run on hardware (e.g., a set of matrix calculations to perform) with an associated set of metrics monitored, typically MIPS/FLOPS (latency) or power consumption. The ML systems community dovetails the ML and hardware communities, which means that a benchmark must function as both.

Benchmarks are important as they provide a means of **standardization** that allows incremental improvements to be clearly demonstrated, as in the case of AlexNet. If every paper was based on a different dataset, using a different set of metrics, it would be difficult to interpret the effectiveness of any given approach. By providing some standardization, we can (1) define a set of rules defining the task, as well as what optimizations are allowed and how they must be implemented (2) help to establish a set of metrics that are meaningful and easily interpreted for the given task, (3) require submitted implementations to be public to promote open-access and reproducibility, and (4) control for elements of randomness that may impact results. This standardization helps to combat some of the problems that are common across the research landscape, such as results that are reported but not public or reproducible, or solutions that are fine-tuned on a specific dataset but not generalizable.

Benchmark Challenges

Since the goals of two different systems (especially in ML) may be different (e.g., optimizing for latency vs. accuracy), different benchmarks are needed for different tasks. Even within the same broad task (e.g., computer vision), there may be applications that may not be exactly the same. For example, an object detection algorithm would require details for each item within the image, as opposed to just a classification value for the whole image as in image classification. Because of this, many benchmarks have been created to cover a variety of different tasks, utilizing different datasets, metrics, and rule sets for reporting purposes.

Some challenges do exist with benchmarks. For example, some benchmarks may be "synthetic", which do not represent real workloads but aim to measure performance relevant to real use cases. This can be problematic, since if the benchmark is not correctly aligned with its associated use case, improvements in the benchmark may not correspond to improved performance in real-world implementations. Naturally, this is also a potential issue if the benchmark is not correctly configured or contains some inherent biases (e.g., imbalanced data classes, specific image orientations) that may not be present in real-world implementations.

Examples

Many benchmarks exist today in both the ML and embedded systems communities. Common benchmarks for ML include: MLPerf inference, MLPerf Mobile, DAWNBench, and AIBench. Benchmarks within the embedded systems community have been around for longer than the ML ones and so have evolved over the years. Nowadays, some of the more common benchmarks include CoreMark (replacing Dhrystone), EEMBC's ULPMark for ultra-low power systems (three variants: ULPMark-CoreMark, ULPMark-CoreProfile, ULPMark-PeripheralProfile) and Embench.

While some other academic benchmarks exist, MLPerf Tiny is currently the only industry-standard TinyML benchmark, which we will go into in more detail in later sections.