TPCx-AI on NVIDIA Jetsons

Robert Bayer, Jon Voigt Tøttrup, and Pınar Tözün

IT University of Copenhagen, Denmark
roba@itu.dk, jvoi@itu.dk, pito@itu.dk

Abstract. Despite their resource- and power-constrained nature, edge devices also exhibit an increase in the available compute and memory resources and heterogeneity, similar to the evolution of server hardware in the past decade. For example, NVIDIA Jetson devices have a system-on-chip (SoC) composed of an ARM CPU and an NVIDIA GPU sharing RAM that could be up to 32 GB. Such an SoC setup offers opportunities to push down complex computations closer to the data source rather than performing them on remote servers.
In this paper, we characterize the performance of two types of NVIDIA Jetson devices for end-to-end machine learning pipelines using the TPCx-AI benchmark. Our results demonstrate that the available memory is the main limitation to performance and scaling up machine learning workloads on edge devices. Despite this limitation, some edge devices show promise when comparing against a desktop hardware in terms of power-efficiency and reduction in data movement. In addition, exploiting the available compute parallelism on these devices can benefit not just model training and inference but also data pre-processing. By parallelizing, we get close to an *order of magnitude* improvement in pre-processing time for one of the TPCx-AI use cases. Finally, while TPCx-AI is a valuable benchmark, it is designed for server settings; therefore, the community needs an end-to-end machine learning benchmark targeting IoT/edge.

Keywords: TPCx-AI · system-on-chip · edge devices · IoT · performance benchmark · resource-aware · resource-constrained · machine learning

1 Introduction

In the past decade, we have seen major advances in the field of machine learning [1]. These advances have been mainly thanks to the availability of powerful hardware and large datasets. On the other hand, today, many data sources are actually small, low-powered edge or Internet-of-Things (IoT) devices, such as mobile phones, micro-controllers in sensors, wearable or self-driving smart platforms, etc. It becomes increasingly important to enable techniques that get more value out of data at these edge points rather than always sending the data to a remote and more powerful hardware device (such as a server in a data center) for further processing and training powerful machine learning models. Getting more value out of data closer to the source would be more secure, create new data-intensive applications at the edge, and enable more cost- and energy-efficient use of data by reducing data movement. However, the challenge is operating on devices that are much more resource-constrained compared to the CPU-GPU co-processor servers that have sparked the machine learning advancements.

Parallel to the evolution of server hardware, the hardware resources available at the edge have also evolved. Such devices have already deployed System-on-Chip (SoC) designs that embraced hardware heterogeneity and co-processors earlier than server hardware. In addition, today, they come with increased memory and higher degree of compute parallelism. For example, NVIDIA offers Jetson devices for edge computing that have an SoC composed of an ARM CPU and an NVIDIA GPU sharing RAM that can be up to 32 GB depending on the device type. This evolution makes moving more data-intensive tasks closer to the sources of data more plausible.

This paper is a step toward understanding the capabilities and limitations of modern edge devices for machine learning pipelines. More specifically, we characterize the performance of two NVIDIA Jetson devices (TX2 and AGX Xavier) compared to a desktop hardware environment using the TPCx-AI standardized benchmark [2]. Both NVIDIA Jetson devices and TPCx-AI are relatively new and unexplored in our community. NVIDIA Jetson devices offer an interesting platform for machine learning at the edge, thanks to the available GPU and main memory. TPCx-AI offers an end-to-end perspective for machine learning, which allows testing the impact of different stages on resource-constrained hardware and the ability to scale the benchmark up and down, which enables stress-testing on devices offering varying compute and memory resources. While MLCommons [3] (formerly known as MLperf [4]) is the older and more mature standardized benchmark collection for machine learning and also offers benchmarks for edge devices and tiny hardware, it misses the end-to-end perspective and the workload scaling aspects of TPCx-AI. This study, therefore, also explores the new TPCx-AI benchmark and its potential when benchmarking modern edge devices. TPCx-IoT [5], which is the standardized TPC benchmark for IoT settings, neither targets AI nor would be easily extended for testing end-to-end machine learning.

The contributions of our performance characterization study is as follows:

1. Despite the increased memory resources at modern edge devices, our results show that the available memory is still the main limitation to performance and scaling up machine learning workloads on edge devices. Despite this limitation, a powerful edge device such as NVIDIA Jetson Xavier show potential to be competitive with a desktop hardware considering its power efficiency and omitted data movement costs.
2. We demonstrate that the increased hardware parallelism at the edge is an important asset that must be exploited. By implementing multi-threaded data pre-processing for one of the TPCx-AI uses cases, we get nearly an order of magnitude improvement in pre-processing time.
3. We find that even though TPCx-AI is an easy-to-deploy benchmark and can give valuable performance insights even on hardware platforms that it is not designed for, the community still needs an end-to-end machine learning benchmark targeting IoT/edge settings.

2 Background

This section describes what resource-aware machine learning constitutes and why it is important (Section 2.1), SoC architectures and their role in resource-aware machine learning (Section 2.2), and the TPCx-AI benchmark (Section 2.3).

2.1 Resource-Aware Machine Learning

Machine learning has become mainstream and is used in all industries, from product recommendation on the e-commerce platform [6] to X-ray classification in the medical industry [7]. The high predictive power of machine learning models, mainly deep learning models, comes at a price; the newer and better models have a higher computational cost [8] and, consequently, higher energy consumption.

While the accuracy of the trained models has traditionally been the primary metric of focus in machine learning, creating more transparency around the computational cost of models has gained traction recently[9]–[11]. This is also partially fueled by the higher focus on the climate crisis motivating researchers to be more resource-aware when designing new machine learning algorithms rather than always increasing hardware - and hence, energy consumption, to increase the model's predictive power. Resource-aware machine learning includes revised algorithms for more effective utilization of general-purpose or specialized hardware or designing specialized hardware for accelerating training and inference [12], [13]. The hardware focus here is not only the server hardware found in data centers but also more resource-constrained hardware found at *the edge*, representing the devices where data is captured such as handheld devices or sensors to collect data about the surroundings.

While the hardware resources on these devices are limited, the workloads are increasingly demanding, making efficiency even more critical. One solution to sidestep this problem is maintaining a steady connection between the data centers and edge devices so that sensor data can be processed quickly in the cloud. However, depending on the data size and proximity to the data center, this can take up too much bandwidth and incur high latency and energy costs. An alternate approach is to perform some or all computations involved in data processing at the place where the data is collected, *at the edge*. This is much more dependent on the capabilities of the device collecting the data, and in this work, our focus is to investigate these capabilities.

2.2 System-on-Chip Devices

System-on-chip (SoC) are devices that integrate all computer components into a single package. These can include CPU, I/O, RAM, and other specialized hardware, such as GPU, wireless radios, or programmable logic. SoCs have become popular with the rise of portable and edge devices and appear in laptops and desktop computers today.

The primary motivator for putting all of the electronics on a single chip is the energy consumption, which is reduced by increasing proximity of components, as energy consumption of data transfers is directly proportional to the distance of the transfer [14]. Furthermore, the higher energy efficiency correlates with thermal efficiency, where the heat dissipated from the SoC devices is much lower than their traditional counterparts. Lastly, the smaller size is also a very appealing feature, mainly for edge devices.

For CPU-GPU co-processors, an SoC design can share RAM between CPU and GPU. This eliminates costly memory transfers between CPU and GPU, leading to lower latency and energy consumption overall. Having memory shared also means that the total memory capacity can be increased, leading to previously unseen capacities for GPUs at the edge, which could alleviate the struggle data scientists meet when trying to train and deploy larger models.

2.3 TPCx-AI Benchmark

TPCx-AI [2] is a benchmark suite developed to test and evaluate the end-to-end machine learning capabilities of a system. The benchmark, which comes with a codebase, provides a platform that

- generates and processes large volumes of data mimicking real-world use cases,
- trains on pre-processed data to produce realistic machine learning models,
- conducts accurate insights for real-world customer scenarios based on the generated models,
- can scale to large-scale distributed configurations, and
- allows for flexibility in configuration changes to meet the demands of the dynamic AI landscape.

For the real-world use cases, the benchmark draws inspiration from the retail industry, where most companies utilize AI techniques to boost their competitiveness. The benchmark size is scalable to accommodate business needs of different sizes. Users can provide, amongst others, a scaling factor (SF), which indicates the size of the data set for benchmarking.

The benchmark consists of ten use cases, each taking advantage of a different machine learning techniques ranging from traditional techniques, such as k-means clustering and naive Bayes classifier, to more advanced and recent techniques, such as recurrent neural networks. The use cases incorporate diverse input data, where tabular data is complemented with image, audio, and textual data; and cover both supervised and unsupervised learning.

Out of the ten use cases only use cases 2, 5, and 9 rely on the use of a GPU. The rest of the use cases rely fully on the use of CPU. The distribution between the amount of data pre-processing and model training/serving varies across the use cases. Use cases 8, 9, and 10 are heavily skewed towards pre-processing. We especially focus on the pre-processing stage of use case 8 in Section 4.3.

Test phases. One key distinguishing factor of TPCx-AI from the older MLCommons benchmark is its focus on end-to-end machine learning instead of solely focusing on training or inference. Therefore, the benchmark comprises six phases, described below, that run sequentially for each use case.

(1) Load test (LT) tests the process of copying the input files generated using the TPCx-AI data generator to the directories they will be fetched from during the benchmark run. This stage tests the storage infrastructure used during the benchmark run.

(2) Power training test (PTT) determines the maximum speed of training phase of each of the use cases, which includes the process to load and pre-process the data before the data is fed to the training process. It outputs (1) the total time of training in addition to the time taken per use case, and (2) the model files, which are to be used in the subsequent test phases.

(3) & (4) Power serving test I & II (PST, PST1 & PST2) determines the maximum speed at which the system-under-test can perform the serving phase of all use cases. Same as PTT, this test includes time to load and pre-process the data before the data is fed to the serving process.

(5) Scoring test performs a separate serving phase of each of the use cases on a newly generated dataset to determine the accuracy or error incurred by each of the use cases.

(6) Throughput test (TT) runs several concurrent streams, each containing all of the use cases in a unique order. This tests the ability of the system-under-test to serve models from different use cases to multiple users. The default value of the number of streams is 2, which can be modified.

Metrics. After every benchmark run, a report is computed by the benchmark utility, providing a set of metrics that can be used to compare the system-under-test with other systems.

The TPCx-AI benchmark defines their own set of metrics [2] [Benchmark Specification, Section 7.5].

$\boldsymbol{T_{ACRONYM}}$ is the geometric mean of the time spent in seconds for each test phase per use case. $ACRONYM$ corresponds to the acronym of the given test phase as listed above. For example, T_{TT} means time spent in *throughput test* phase of the benchmark.

AIUCpm@SF is **AI U**se **C**ase **p**erformance **m**etric **at S**caling **F**actor and summarizes a subset of the other performance metrics in a single scalar value. It is computed by the following formula, where SF is the scaling factor and N is the number of use cases (always 10):

$$\frac{SF \cdot N \cdot 60}{\sqrt[4]{T_{LT} \cdot T_{PTT} \cdot T_{PST} \cdot T_{TT}}}$$

DATAGEN is the time spent in seconds generating the complete data-set.

\$/AIUCpm@SF is a metric to highlight price per performance. For Transaction Processing and Performance Council (TPC), it is customary to report the total cost of a system divided by the achieved performance [15]. This is also a useful metric that gives a rough estimate of how much value for money that a system yields. In our experiments, the performance metric is AIUCpm@SF and the currency is USD.

Except for $AIUCpm@SF$, lower values are better for these metrics.

3 Related Work

TPCx-AI is a relatively new standardized benchmark [16]. Hence, to the best of our knowledge, our work is the first study that utilizes TPCx-AI for performance characterization of hardware, especially edge devices, since the introduction of TPCx-AI in [17].

In contrast, several works have tested edge devices for artificial intelligence. In [18], an NVIDIA Jetson Nano, a Raspberry Pi 4, a Google Coral Dev Board and an Arduino Nano 33 BLE microcontroller are benchmarked on selected deep learning tasks focusing on training and inference. In [19], a benchmark suite is developed targeting machine learning and cognitive science applications to test cloud, edge, and mobile devices. The benchmark is also adapted to test distributed computers serving multiple end-users. In addition to working on benchmarks, in [20], the authors focus on model instability at the edge.

Our work is complementary to these works since our aim is to study the performance of modern powerful edge devices such as NVIDIA Jetsons for end-to-end machine learning tasks (not just inference or training). In parallel, we are investigating the potential of the TPCx-AI benchmark beyond benchmarking on server hardware.

4 Experimental Methodology and Setup

Our goal is to characterize the performance of modern, powerful edge devices for end-to-end machine learning. This section presents the experimental methodology and setup we deploy to achieve this goal.

4.1 Systems

To represent state-of-the-art modern edge devices, we pick two offerings from NVIDIA with varying hardware resources. In addition, we use a desktop hardware setup as the baseline to compare against the edge devices. Section 4.1 summarizes the specifications of these devices.

NVIDIA Jetson TX2, labeled `TX2` in short, is a portable SoC composed of a power-efficient ARM-CPU & NVIDIA GPU, designed for embedded systems that require GPU-friendly computations, e.g., image processing, video encoding/decoding, and machine learning tasks. TX2 used in our experiments comes with 8 GB RAM (shared between CPU & GPU) and 32 GB eMMC storage, a 6-core $\sim$ 2 GHz ARM64 CPU and 256 CUDA Core GPU [21], [22]. The device can operate at a wattage between 7.5 W-15 W. We configured our TX2 to operate at 15 W, increasing the maximum clock rates but doubling power consumption.

We also added extra storage in the form of an SD-card (32 GB) to fit the entire TPCx-AI benchmark suite.

To accommodate the memory requirements of TPCx-AI scaling factors 1-3, we reserved 12 GB of disk space as swap memory, in addition to the default of $\sim$ 4 GB compressed swap memory (`zram`).

NVIDIA Jetson AGX Xavier, labeled `Xavier` in short, is also a portable computer by NVIDIA, but with more powerful hardware and designed specifically for autonomous machines [23]. Xavier used in our experiments has an 8-core $\sim$ 2.2 GHz ARM64 CPU, a 512 CUDA-core GPU and $\sim$ 32 GB memory shared between the CPU and GPU. The device can operate at a wattage between 15 W-30 W. As with TX2, we configured the device to operate at the maximum wattage (30 W) to maximize clock rates despite the double power consumption. Like TX2, the device is fitted with 32 GB eMMC storage [21], but unlike TX2, we opted to add a USB 3.0 storage device (64 GB).

Software-wise, both TX2 and Xavier runs NVIDIA's Ubuntu distribution for the Jetsons, as provided by NVIDIA Jetpack [24] on both devices. We also used Jetpack to install the most commonly used machine learning libraries such as CUDnn and OpenCV.

Desktop setup, also labeled as `Desktop`, that we used as a baseline for our comparisons includes an NVIDIA GeForce RTX 2070 GPU and an x86 6th gen Intel i7 CPU with 16 GiB RAM. The power consumption of the device is estimated from the power requirements of the GPU to be at most 550 W [25].

As a rule of thumb, the power consumption can be assumed to be about one order of magnitude above that of a Jetson device.

Device	GPU	CPU	RAM	PWR	Price
TX2	NVIDIA Pascal, 256 CUDA Cores	NVIDIA Denver (2 Cores) & Arm Cortex A57 (4 Cores) @ 2.0 GHz	8 GB	15W	$399
Xavier	NVIDIA Volta, 512 CUDA Cores, 64 Tensor Cores	8 Cores ARM v8.2 64-bit @ 2.2 GHz	32 GB	30W	$699
Desktop	NVIDIA RTX 2070, 2304 CUDA Cores, 288 Tensor Cores	8 Cores Intel Core i7-6700K @ 4.0 GHz	16 GB (CPU), 8 GB (GPU)	$\sim$ 550W	-

Table 1: Systems-under-test in our experiments. The information for the Jetson devices are taken from [21], [26]. The *price* column represents manufacturer suggested retail price, which we couldn't find for all the Desktop components.

4.2 Metrics

The metrics used to reason about the performance of Jetsons is organized into three categories: application-level (reported by TPCx-AI), hardware utilization, and power consumption metrics.

Application-level. These are the metrics reported by TPCx-AI (see Section 2.3). We omit the `DATAGEN` metric due to the added network overhead overshadowing the actual data generation time in our setup (see Section 4.3).

Hardware utilization. While the benchmark metrics show how quickly the system completes different AI-related tasks, it leaves the question of how *efficiently* the resources of this system are utilized while performing these tasks. Understanding hardware utilization characteristics can also help comprehend performance differences across the benchmark use cases and hardware systems.

To measure hardware utilization, we monitor the CPU and GPU utilization and memory consumption, at each second using the `tegrastats` utility provided by NVIDIA for Tegra-chipset [27], which is what the Jetson devices have. On the desktop hardware, we use `nvidia-smi` [28] and `ps` [29] utility provided by NVIDIA and unix, respectively, for the same measurements.

Power consumption. In addition to the hardware utilization, we record the power consumption of each device to explore efficiency further. For the NVIDIA Jetson devices, the system power consumption is collected by the `tegrastats` utility. However, there is no way to collect the complete information through the available software for the desktop. In that case, we only collect the GPU's power consumption reported by `nvidia-smi`. The power consumption is recorded as a snapshot measurement of system wattage in both cases. To complement this, we accumulate the power consumption over time as a single watt-hour measurement for each use case of each benchmark run.

4.3 Benchmark Suite Modifications

We made three modifications to the TPCx-AI benchmark suite[1]: (1) to run the benchmarks in a non-x86 environment, (2) bug fix related to the batch size in Use Case 5, and (3) a performance-related modification.

First, the data-generation component of the TPCx-AI benchmark suite (the `PDGF`[2, Specification p. 64]) is compiled for x86-64 architectures only and cannot run on our Jetson devices. Therefore, we set up a separate Intel x86-64 machine with a purpose-built HTTP server to run the data-generation software and provide the generated data files. On our Jetson devices, we replaced the PDGF-executable with an executable that acts as a client for the HTTP server. This way, we move the data from the x86 machine to the Jetsons over the network before the benchmarking phase starts.

Then, we discovered a bug in Use Case 5 (line 166), where a parameter `batch` was not being passed to the `serve` function, leading to much higher memory

[1] The changes we made to the codebase can be found at `https://github.com/ContainedBlargh/TPCx-AI-on-Nvidia-Jetsons`

demands for serving than for training. As that parameter is configurable but not passed, we assumed this was a bug.

Finally, we changed the data pre-processing step of Use Case 8 to be multi-threaded, which improved the times to run this use case drastically. In Section 5.3, we highlight the performance impact of this modification.

5 Results

We ran the TPCx-AI benchmark five times and reported the mean and standard deviation for each configuration (device & scaling factor). As for the scaling factor, we use the values of 1 and 3, as higher scaling factors cannot fit in Jetson devices. Unless stated otherwise using the label `original`, all reported results are with the modified version of use case 8. The results are split into three parts: (1) overall results from the whole benchmark run, (2) time-breakdowns for each use case, and (3) impact of parallelizing pre-processing of use case 8.

Metric		TX2, SF=1	Xavier, SF=1	Xavier, SF=3	Desktop, SF=1	Desktop, SF=3
AIUCpm@SF	Mean	8.18	28.91	31.10	90.15	152.29
	St. dev.	2.54	1.48	3.80	2.42	5.19
$/AIUCpm@SF	Mean	37.84	24.23	22.73	-*	-*
	St. dev.	9.10	1.29	2.66	-*	-*
T_{LT} (seconds)	Mean	39.34 s	1.85 s	27.72 s	1.33 s	2.33 s
	St. dev.	19.81 s	0.40 s	13.57 s	0.04 s	0.25 s
T_{PTT} (seconds)	Mean	308.19 s	94.66 s	198.88 s	34.91 s	84.17 s
	St. dev.	7.10 s	3.27 s	5.66 s	1.45 s	3.53 s
T_{PST} (seconds)	Mean	48.51 s	28.06 s	45.49 s	6.99 s	10.28 s
	St. dev.	1.42 s	0.67 s	2.19 s	0.38 s	0.09 s
T_{TT} (seconds)	Mean	72.89 s	38.52 s	50.91 s	6.06 s	9.75 s
	St. dev.	4.10 s	1.65 s	3.12 s	0.34 s	0.13 s

Table 2: TPCx-AI metrics (see Section 2.3 for details). *Due to unknown price, the $/AIUCpm@SF has been omitted.

5.1 Whole Benchmark Run

Table 2 lists the mean and standard deviation values for the metrics reported by TPCx-AI. To help us explain some of these benchmark metrics, Table 3

Total swapped	TX2, SF=1, original	TX2, SF=1	Xavier, SF=1	Xavier, SF=3	Desktop, SF=1	Desktop, SF=3
Mean (MB)	4271.8	3974.6	0.30	1008.99	3597.33	19335.2

Table 3: Total memory (MB) swapped to disk during training of all use cases.

presents the amount of memory swapped during the whole benchmark run for each device. When we monitored the hardware utilization information throughout the benchmark runs, it became clear that memory was the primary resource under pressure. All the hardware devices used in this work have relatively limited physical memory (8 GB to 32 GB) compared to today's state-of-the-art server hardware in data centers (100 GB to TBs). As a result, when the working dataset size of TPCx-AI does not fit into the available device memory, it triggers *memory swaps*, meaning parts of the working memory are moved to the disk so that currently required data can be loaded into memory. Thus, instead of showing CPU/GPU utilization, we have results for swapped memory here.

Looking at the benchmark's performance summary metric, *AIUCpm*, `Desktop` hardware is an order of magnitude better than `TX2`. As Table 3 demonstrates, `TX2` exhibits a high number of swaps due to its smaller memory size, which results in this behavior. In addition, the SD card in `TX2` causes a high T_{LT}, which is the most data-intensive task, since it includes reading/writing data from persistent storage.

Comparing `Xavier` to `Desktop` hardware, the difference in performance increases with a higher scaling factor. However, `Xavier` performs roughly the same in terms of *AIUCpm* across the two scaling factors showing that it scales well with the increase in scaling factor. In addition, it has larger main memory and exhibits the lowest number of swaps. However, it has two main bottlenecks compared to `Desktop`, which results in lower performance. First, `Xavier` has a slower ARM processor compared to the x86 in `Desktop`. Second, the eMMC-based storage is also slower than the SATA SSD in `Desktop`, as highlighted by the different behavior between the T_{LT} results for the two scaling factors.

We leave the parameters that determine the degree of parallelism to default values set by TPCx-AI, except for the pre-processing phase of use case 8 (see Section 5.3). Looking at the results for T_{PTT}, T_{PST}, T_{TT}, we see the impact of both the differences in processor speed and the different degrees of hardware parallelism each device provides.

From a price/performance perspective, *\$/AIUCPm*, both Jetson devices perform similarly.

Lastly, Table 4 presents the total power consumption for each device for the whole benchmark run. Despite the lower power draw of `TX2` compared to the other devices (Section 4.1), it consumes the highest power to complete the benchmark since it takes a longer time to complete a benchmark run. `Xavier`

Power consumption	TX2, SF=1, original	TX2, SF=1	Xavier, SF=1	Xavier, SF=3	Desktop, SF=1	Desktop, SF=3
Mean (Wh)	30.31	19.09	8.24	16.27	10.23	12.50
St. dev (Wh)	0.58	0.34	0.21	0.78	1.21	0.32

Table 4: Total power consumption by device across all benchmarks in watt-hours. Note that `Desktop` power measurements are GPU-only.

behaves similarly to `Desktop` even though the `Desktop` results only count the GPU power consumption. Based on this, `Xavier` is competitive in terms of power/performance ratio over the other devices.

5.2 Time-breakdown per Use Case

Figure 1 and Figure 2 breaks the training (Phase #2) and serving (Phase #3 & #4) phase (Section 2.3) of each use case in the benchmark, respectively. TPCx-AI reports time (T_{PTT} and T_{PST}) at a coarse granularity from these phases for the individual use cases. We further break this time into *loading*, which is the time to fetch data from storage and join/merge them if there are multiple data files, *pre-processing*, which is the time to pre-process the data to be fed to training/serving, and *training/serving*, which is the time that goes to actual training/serving.

The figures show that `TX2` takes the most time and `Desktop` hardware takes the least time to complete the phases. This is expected considering the hardware resources available on these devices and the results reported in Section 5.1. Furthermore, we observe that different use cases stress the different phases of training and serving. Use case 1 exhibits higher loading times, while training/serving times dominate the time-breakdown for use cases 2, 4, 5, 6, and 7. On the other hand, for use cases 8, 9, and 10, the pre-processing times are more pronounced (as also Section 2.3 highlights). This shows that TPCx-AI use cases exhibit a good variety in terms of the machine learning tasks they stress at runtime.

Comparing the results of scaling factors (SF) 1 and 3, there is no direct scaling of runtimes for different components. For some use cases, the times roughly triple, but in most cases, they are sub- or super-scalar. We omit SF=3 results for `TX2` due to its prohibitively long run times, which are a result of the low memory (8 GB) on this device causing frequent swap operations (Table 3).

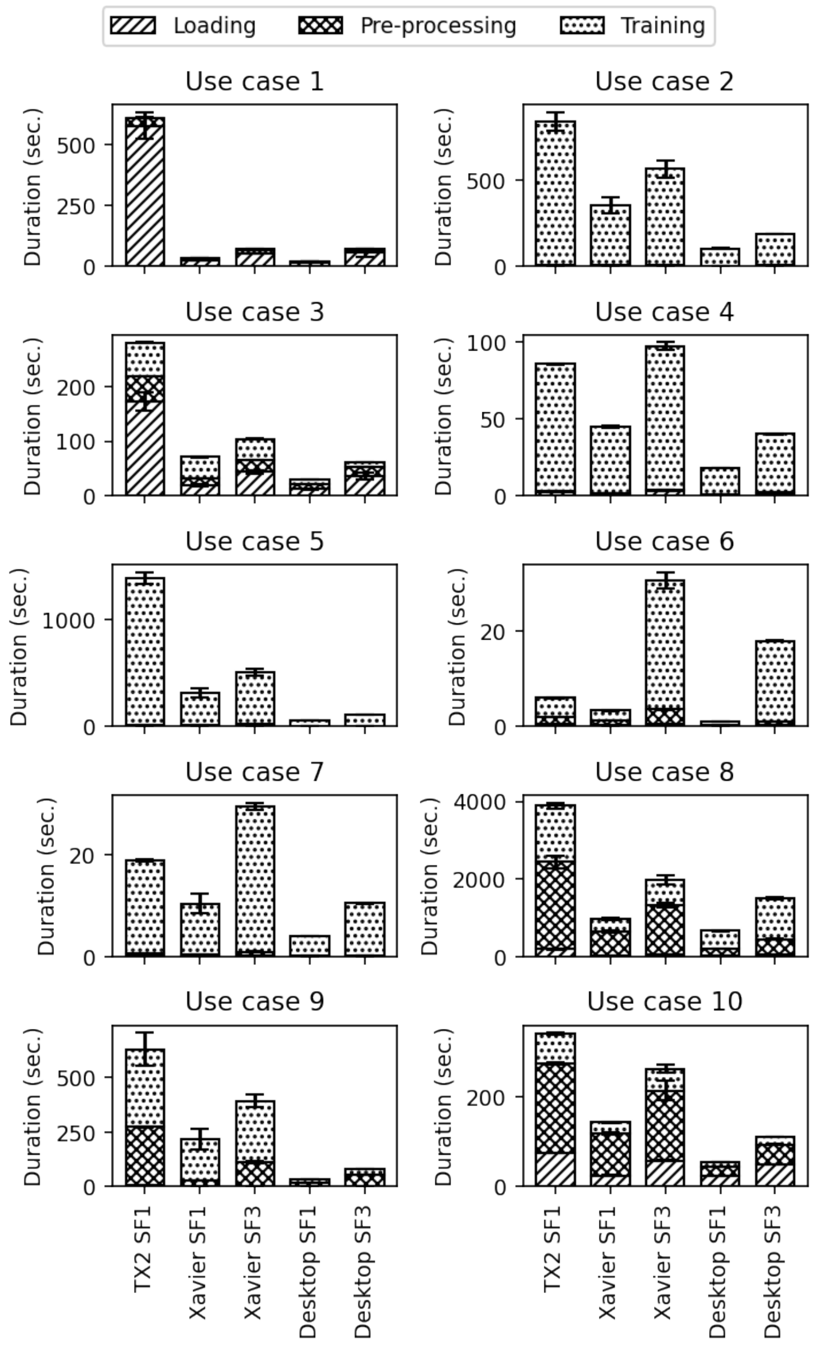

Fig. 1: Time-breakdown of the training phase of individual TPCx-AI use cases using revised use case 8. Notice the different scale on y-axes.

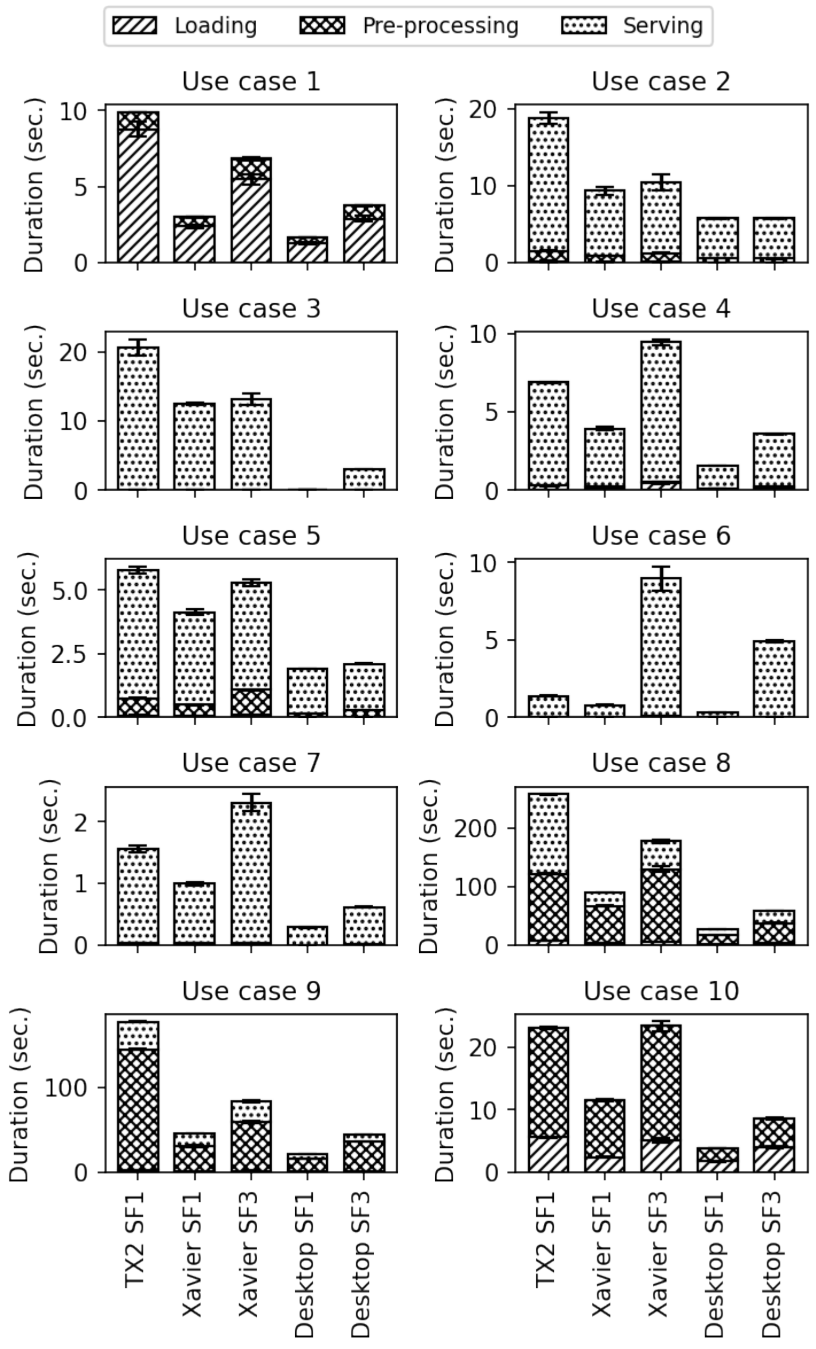

Fig. 2: Time-breakdown of the serving phase of individual TPCx-AI use cases using revised use case 8. Notice the different scale on y-axes.

Power consumption	TX2, SF=1, original	TX2, SF=1	Xavier, SF=1	Xavier, SF=3	Desktop, SF=1	Desktop, SF=3
Mean (Wh)	25.39	14.27	3.66	7.38	2.86	1.41
St. dev (Wh)	0.50	0.46	0.10	0.56	0.12	0.25

Table 5: Power consumption for use case 8 by device in watt-hours. Note that `Desktop` power measurements are GPU-only.

5.3 Use Case 8

When we first ran the whole TPCx-AI benchmark suite on `TX2` and `Xavier` out of the box, after setting up both environments with necessary software libraries and additional storage, it took about 4.5 and 2 hours, respectively. The time-breakdowns for individual use cases showed that the pre-processing time of the training phase of use case 8 was a clear outlier. Digging deeper into its code, we realized that it could be parallelized with slight modifications. After applying our changes and running this pre-processing step using three threads, the total run times of the whole benchmark run were reduced to about 2.75 hours for `TX2` and 45 minutes for `Xavier`. Focusing solely on this pre-processing step, the time to complete it got reduced by 70% and 84% on `TX2` and `Xavier`, respectively.

Comparing the two variations of the `TX2` SF=1 results on Table 3 and Table 4, where *original* refers to TPCx-AI code without the use case 8 modifications, also shows the positive impact of this code change in terms of reducing both the overall memory pressure and power consumption. Table 5 reports the power consumption for use case 8, specifically showing a similar impact.

6 Discussion

This section discusses the highlights of the results in Section 5 and pros/cons of using TPCx-AI to benchmark edge devices for end-to-end machine learning.

6.1 Machine Learning on Jetsons

Our goal was to characterize the performance of NVIDIA Jetson devices for machine learning. Our results highlight that main memory can easily become the factor that limits performance, especially on the smaller Jetson devices, such as `TX2`. On one hand, the more powerful processors, like the one we have on `Desktop`, can compensate for the memory bottleneck. On the other hand, from a cost- and energy-efficiency perspective, the more expensive and power-hungry `Xavier` device has an advantage over `TX2` since it can complete the machine learning tasks faster. `Xavier` performs even at a similar level with `Desktop` when it comes to energy-efficiency.

In a data center setting, it is common wisdom [30] that if an application is latency-critical, a fast multicore x86 processor, like the one on `Desktop`, is more performance-efficient than an ARM processor, like the one on `Xavier` and `TX2`. Our results also corroborate this wisdom if we assume that all the processing is done locally omitting the cost of going over the network. More specifically, the Jetson devices are designed for the edge. In addition to being designed for energy-efficiency, they are also deployed at the source of data collection. In contrast, `Desktop` would be a device where the data is sent to for further processing from its source device. Therefore, when considering latency in an edge setting, one must take into account the cost of moving the data from the source to another device if the processing cannot be done at the edge. From this perspective, declaring a clear winner between `Xavier` and `Desktop` for latency is challenging.

6.2 TPCx-AI Benchmark for Edge Devices

TPCx-AI is a relatively new standardized benchmark and has not been designed with edge/IoT settings in mind. It targets end-to-end machine learning at data center or high-performance computing settings, where both compute and memory resources are plenty. Such settings can handle the memory pressure and throughput needs of TPCx-AI easily. In addition, the reference TPCx-AI implementation, while tremendously helpful for using the benchmark almost out of the box, is there to guide people to deploy the same use cases on the machine learning framework they want to analyze.

In our work, we use TPCx-AI on edge devices, which are designed to minimize latency at small-scale rather than throughput, i.e., work quickly with small-scale data, and to be energy-efficient. We also aimed at using TPCx-AI's reference implementation as is. One can claim that our setup is not an ideal use of this benchmark. However, for testing end-to-end machine learning, to the best of our knowledge, there is no other viable option. In addition, the ability to scale the workloads up and down using scaling factors, which is a common functionality in TPC benchmarks, is very valuable for testing hardware with differing resource characteristics. Thus, testing the potential of this benchmark at the edge settings was worthwhile to us.

Overall, we found that using the python-based reference implementation; one can use TPCx-AI *almost* out of the box, even for edge settings. The main challenge against using it directly out of the box was the data generation component, which is closed source and built for x86 environments. Thus, we had to generate our data on an x86 machine and move our data to the edge device (Section 4.3). For the python libraries, using the counter-parts available on the Jetson was relatively straightforward. While all this still requires non-negligible setup time, it was feasible since it took us a couple weeks, not months.

On the other hand, even the TPCx-AI scale factor 1 is too big for the memory resources of the smallest device we had, `TX2`. In addition, the nature of the throughput tests where multiple models were in use may not be representative for edge settings where typically a few models would be deployed. Furthermore, considering the higher availability of CPU parallelism and GPU resources at the edge, having more parts of the reference implementation that can exploit CPU parallelism or run on a GPU would be helpful. Finally, when testing hardware, especially at the edge, both cost- and power-efficiency must be considered in addition to the *AIUCpm*. Although TPC also has a standardized way to measure power/performance trade-off, it is rarely reported in the results published on TPC's webpage. Considering our results and increasing importance of being power-efficient, we argue that more people should publish power results.

Going forward, when designing an end-to-end machine learning benchmark, it is preferable if the reference implementation can scale the workload up and down similar to TPCx-AI but simultaneously have the ability to scale the datasets further down and exploit more and different types of hardware parallelism. While doing this design, the use cases and throughput tests should be adjusted accordingly. For example, one can include a *transfer learning* (partial training on pre-trained models) use case since edge devices today are rarely used for the full training of the models. On the other hand, having a few full training cases would be interesting to observe the strengths of a particular device.

Lastly, one can also devise an experimental study by picking and choosing different combinations of use cases instead of running the full benchmark suite. While this was possible for TPCx-AI, we have not investigated this route yet.

7 Conclusion

In this work, we performed a performance characterization study of two modern high-end edge devices, NVIDIA Jetson TX2 and AGX Xavier, for end-to-end machine learning. We identified the TPCx-AI benchmark as a good candidate for generating the workload for such a study and a consume-grade CPU-GPU co-processor, desktop machine, as a baseline. Our study shows that the small memory of TX2 is a limiting factor of performance, while Xavier achieves a good cost- and energy-efficiency. In addition, exploiting the increasing degrees and types of hardware parallelism is not only crucial for big server settings but also for the edge. Finally, while TPCx-AI provided us with many valuable insights without high deployment cost, a more thorough characterization of such edge devices requires a benchmark that is specialized for end-to-end machine learning for the edge/IoT settings.

Acknowledgements

The work is funded by the Independent Research Fund Denmark's (Danmarks Frie Forskningsfond; DFF) Inge Lehmann program under grant agreement number 0171-00062B and the European Union's Horizon 2020 research and innovation program under grant agreement number 957407. We further thank DASYA lab members at IT University of Copenhagen for their support.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *NIPS*, 2012, pp. 1097–1105.

[2] Transaction Processing Performance Council (TPC), *tpc express ai (tpcx-ai) standard specification revision 1.0.0*.

[3] MLCommons, *MLCommons*, `https://mlcommons.org/en/`.

[4] P. Mattson, C. Cheng, G. Diamos, *et al.*, “MLPerf Training Benchmark,” in *MLSys*, 2020, pp. 336–349.

[5] Transaction Processing Performance Council (TPC), *tpc express iot (tpcx-iot) standard specification revision 2.0.1*.

[6] J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, and D. L. Lee, *Billion-scale commodity embedding for e-commerce recommendation in alibaba*, 2018.

[7] D. S. Kermany, M. Goldbaum, W. Cai, *et al.*, “Identifying medical diagnoses and treatable diseases by image-based deep learning,” *Cell*, vol. 172, no. 5, pp. 1122–1131, 2018.

[8] OpenAI, *AI and Compute*, `https://openai.com/blog/ai-and-compute/`, 2018. (visited on 03/31/2022).

[9] E. Strubell, A. Ganesh, and A. McCallum, “Energy and Policy Considerations for Deep Learning in NLP,” in *ACL*, 2019, pp. 3645–3650.

[10] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, “Green AI,” *CACM*, vol. 63, no. 12, pp. 54–63, 2020.

[11] D. Patterson, J. Gonzalez, Q. Le, *et al.*, *Carbon Emissions and Large Neural Network Training*, 2021.

[12] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, “Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks,” *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.

[13] T.-J. Yang, Y.-H. Chen, and V. Sze, “Designing energy-efficient convolutional neural networks using energy-aware pruning,” in *CVPR*, 2017.

[14] V. Adhinarayanan, I. Paul, J. L. Greathouse, W. Huang, A. Pattnaik, and W.-c. Feng, “Measuring and Modeling On-Chip Interconnect Power on Real Hardware,” *IISWC*, pp. 1–11, 2016.

[15] Transaction Processing and Performance Council, “TPC Express Benchmark™ AI - Full Disclosure Report,” 2022.

[16] N. Ihde, P. Marten, A. Eleliemy, *et al.*, “A Survey of Big Data, High Performance Computing, and Machine Learning Benchmarks,” in *TPCTC*, 2021, pp. 98–118.

[17] T. Rabl, C. Brücke, P. Härtling, *et al.*, “ADABench - Towards an Industry Standard Benchmark for Advanced Analytics,” in *TPCTC*, 2019, pp. 47–63.

[18] S. P. Baller, A. Jindal, M. Chadha, and M. Gerndt, “Deepedgebench: Benchmarking deep neural networks on edge devices,” pp. 20–30, 2021.

[19] T. Hao, K. Hwang, J. Zhan, Y. Li, and Y. Cao, “Scenario-based AI Benchmark Evaluation of Distributed Cloud/Edge Computing Systems,” *IEEE ToCs*, pp. 1–1, 2022.

[20] E. Cidon, E. Pergament, Z. Asgar, A. Cidon, and S. Katti, “Characterizing and taming model instability across edge devices,” in *MLSys*, pp. 624–636.

[21] NVIDIA, *Jetson Modules*, https://developer.nvidia.com/embedded/jetson-modules, 2021. (visited on 02/10/2022).

[22] ——, *Jetson TX2 Module*, https://developer.nvidia.com/embedded/jetson-tx2, 2021. (visited on 02/10/2022).

[23] ——, *Jetson AGX Xavier Modules*, https://developer.nvidia.com/embedded/jetson-agx-xavier, 2021. (visited on 02/10/2022).

[24] ——, *JetPack SDK*, https://developer.nvidia.com/embedded/jetpack, 2021. (visited on 02/10/2022).

[25] ——, *RTX 2070*, https://www.nvidia.com/en-me/geforce/graphics-cards/rtx-2070/, 2022. (visited on 02/28/2022).

[26] ——, *Jetson FAQ — NVIDIA Developer*, https://developer.nvidia.com/embedded/faq#jetson-prices, 2021. (visited on 05/31/2022).

[27] ——, *Tegrastats Utility*, https://docs.nvidia.com/drive/drive_os_5.1.6.1L/nvvib_docs/index.html#page/DRIVE_OS_Linux_SDK_Development_Guide/Utilities/util_tegrastats.html, 2021. (visited on 02/23/2022).

[28] pmav99, *Nvsmi*, https://github.com/pmav99/nvsmi, 2022. (visited on 03/03/2022).

[29] G. Rodola, *Psutil*, https://github.com/giampaolo/psutil, 2022. (visited on 03/03/2022).

[30] U. Hölzle, "Brawny cores still beat wimpy cores, most of the time," *IEEE Micro*, pp. 23–24, 2010.