Don't Miss the Train: A Case for Systems Research into Training on the Edge

Prashanthi S.K, Aakash Khochare, Sai Anuroop Kesanapalli, Rahul Bhope, and Yogesh Simmhan
Indian Institute of Science, Bangalore 560012, India
Email: {prashanthis, aakhochare, saiak, rahulbhope, simmhan}@iisc.ac.in

I. Introduction and Motivation

The rapid growth of Internet of Things (IoT) and autonomous systems has led to the deployment of *edge devices* close to the sensing data source for low-latency computation. Besides data pre-processing and movement to the cloud, edge devices ranging from Raspberry Pis to Nvidia Jetson with low-power ARM processors are also used for *inferencing* tasks using machine learning and Deep Neural Network (DNN) models for domains like speech, computer vision and Augumented/Virtual Reality [1]. There has also been *systems research* into optimizing the inferencing performance on such devices, including identifying optimal device configurations to minimize latency, power and energy [2], and partitioning an inferencing workload across heterogeneous devices [3].

Edge devices, however, are becoming more powerful by the day while retaining their low power envelope, as is evident in Figure 1 for Nvidia Jetson, Google Coral and Intel Movidius *edge accelerators* in the last 7 years. E.g., Nvidia's AGX Orin will have 12 ARM Cortex A78AE CPU cores at 2 GHz and a GPU with 2048 CUDA cores and 64 Ampere tensor cores to deliver 200 TOPS of performance. With 64 GB of RAM shared between CPU and GPU, this device is comparable to an RTX 3060 Ti GPU workstation but with $\leq$ 50 W of peak power and measuring 100mm x 87mm. Therefore, we posit that *accelerated edge devices are becoming competitive for training DNN models* as well, co-locating the compute with local training data. This includes model training on a single edge, as well as federated learning that is gaining prominence.

The current research into edge training is limited. In our review of 77 papers on federated learning published between 2016–2021, only 19% of them use accelerated edge devices and only 3 papers consider any systems research aspects. Despite similarities, GPU edge devices have key hardware differences compared to GPU servers, such as the use of shared memory between CPU and GPU, low-power but slower LPDDR memory, and diverse storage media like SD card, eMMC and NVMe SSD. Energy minimization tends to be a goal. Also, unlike latency-sensitive inferencing which only involves the forward pass on the DNN, training workloads are long running, use more memory and disk accesses due to large training datasets, and involve heavier computation due to the backward pass and iterative nature of training. Lastly, training stresses all layers of the hardware – disk, memory, CPU, and GPU, and the weakest link can be the bottleneck.

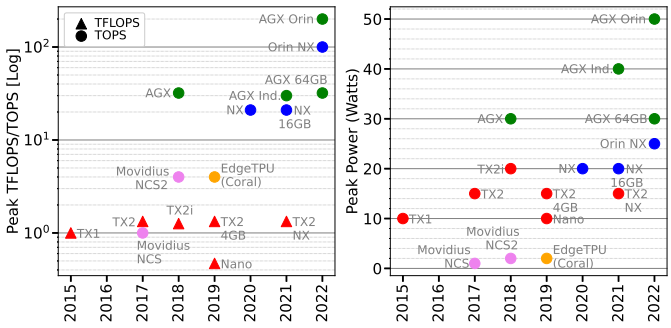

Figure 1: Compute & power trends of accelerated edges

In this position paper, we explore the systems research challenges offered by DNN training on accelerated edge devices. Specifically, we examine the Nvidia Jetson devices, and explore system tuning opportunities offered by frequency and power controls, workload-specific hardware configurations, utilizing shared memory effectively, minimizing data stalls, etc. Our observations are based on first-hand experience with 20 Nvidia Jetson devices – Xavier AGX, Xavier NX, Nano, TX2 and TX1 development kits which have 2-8 ARM cores at 1.43-2.26 GHz, and a peak power of 10W–30W.

II. Systems Research Challenges

1) Tuning frequency and power: Edge devices expose a number of pre-defined *power modes*, with different mixes of CPU, GPU and memory frequencies, peak power and number of active CPU cores. In the Jetsons, users can also define custom power modes with fine-grained control over these. *Dynamic Voltage and Frequency Scaling (DVFS)* dynamically alters the frequency of the GPU, CPU and memory to save power. This can be disabled and the power-mode frequencies set to the maximum to obtain the best performance.

However, given the wide parameter space, picking the sweet-spot for such energy–latency trade-offs is challenging. There can also be intelligence to dynamically switch frequencies and power modes for different phases of the training workload; any overheads for switching [2] are amortized over the longer training periods. Further, from our experiments, we also observe an interplay between these parameters. E.g., when training a small model like LeNet, the time for GPU computation is affected by the CPU frequency. The latter handles the GPU kernel launch, and the launch overheads are significant compared to the kernel computation on the GPU. So, there are research opportunities in detailed modeling of

the edge devices and characterization of deep learning (DL) workloads for intelligent system configuration and scheduling.

2) Stalls in pipelined training: The typical training pipeline has 3 stages that execute for every mini-batch in the epoch – fetch data from disk, pre-process on CPU, and train on GPU. DL frameworks like PyTorch allow these stages to pipeline across mini-batches. However, pipelining benefits are diminished on GPU servers if the system is not balanced and GPU is often idle, *stalling* on the slower fetch and pre-process stages [4]. We observe similar stalls when training on edge devices, but they offer unique solutions.

A basic optimization is to enable pipelining in the DL framework, e.g., by assigning separate workers for the fetch and pre-process in PyTorch. Assigning multiple workers for these stages may not necessarily offer more benefits. Here, the relative performance of the storage media, CPU and GPU come into play. From our experiments, we notice that for faster storage like NVMe SSD, pipelining with a single worker is sufficient to eliminate GPU stalls. But slower HDD needs more workers for concurrent fetches. This also means that a slower cheaper/higher capacity storage device, with more workers, can match the performance of a faster costlier device. Stall can also be hidden if there is sufficient memory to fit a large part of the training data across epochs. We see that the OS serves the data from the page cache. So, careful hardware choices can save capital costs for large edge deployments.

3) Implications of shared memory: Jetson edge devices share the RAM between CPU and GPU. This can provide more RAM to the GPU – a Xavier AGX 64 GB has more memory than even server-grade GPUs, albeit a slower LPDDR one. This can also mitigate data movement costs from CPU memory to GPU memory. However, due to the way memory is managed by CUDA libraries, there may still be a copy involved. This offers opportunities for systems research on memory management by accelerator libraries. Recent systems advances like unified memory can offer potential improvements, though they currently suffer from data initialization overheads [5].

However, the more memory that the GPU uses, the less is available for CPU-based tasks. With larger training models that occupy more RAM, the OS page cache retains less of the training data and this increases the fetch stalls. The default caching policy in Linux, which relies on temporal locality, is not well-suited for DL workloads where the same data is accessed across epochs, but in different order. This offers opportunities for alternate DL-sensitive caching strategies.

4) Device variability: A recent paper [6] indicated that Jetson edge devices exhibit significant variability in inference latency and power drawn across different instances of the same device type. This can prevent stable performance for inference tasks, and also affect reproducibility for systems research. However, unlike DNN inferencing that runs in milliseconds, training happens over minutes or hours and is less affected by fine-grained variability. In our experiments with training MobileNet on 3 AGX and 4 NX devices, the training latency per epoch was consistent across device instances, with a coefficient of variance $< 1\%$. However, we observed significant variability in training time with changes in the software setup, e.g., OS and PyTorch versions. This requires more careful analysis to get a conclusive outcome. In this regard, standard DNN benchmarks like MLPerf, which have profiles for training on workstations and HPC servers, but only inferencing on the edge, need to be extended to profiles for training on accelerated edges too. This can help uniformly compare the performance across device types and instances.

5) Virtualization: Virtualization and containerization can help with application packaging, resource and application sandboxing, and multi-tenancy. Edge device types like the Jetson Nano are resource constrained and containerization using Nvidia's JetPack and Docker will balance isolation and performance. However, more powerful devices like the AGX can benefit from virtualization through improved resource utilization using multi-tenancy, e.g., training different models on the same data. The Carmel cores of AGX support ARM Virtualization Host Extensions (VHE) for hardware-assisted CPU and memory virtualization. However, GPU virtualization on the edge only supports serial time-shared access to the GPU. A detailed study of the performance overheads of virtualized or containerized training is warranted. In the absence of true GPU virtualization, there is opportunity for research into edge middleware that can schedule and overlap training across containers or VMs while serializing access to the shared GPU.

III. Conclusions

In this paper, we have argued the need for sustained systems research into fully exploiting the potential of accelerated edge devices for DNN training, to balance power, performance and cost. While we limited our exploration to optimized training opportunities on a single Jetson-class edge for brevity, there are many such opportunities on other edge accelerators like Google Coral and Intel Movidius, and for distributed and federated learning on such edge devices. Distributed training frameworks can allow resource consolidation of disparate and heterogeneous edge devices in a smart city for opportunistic computing, and balancing performance, power and cost. Federated learning poses unique challenges of device selection to minimize and synchronize training time across devices. Accurate single-system modeling of an edge can help give deterministic training time estimates for local training.

References

[1] M. G. S. Murshed *et al.*, "Machine learning at the network edge: A survey," *ACM Comput. Surv.*, vol. 54, no. 8, 2021.

[2] H. A. Abdelhafez and M. Ripeanu, "Studying the impact of CPU and memory controller frequencies on power consumption of the Jetson TX1," in *IEEE Intl. Conf. on Fog and Mobile Edge Comp. (FMEC)*, 2019.

[3] L. Zeng, X. Chen, Z. Zhou, L. Yang, and J. Zhang, "CoEdge: Cooperative DNN inference with adaptive workload partitioning over heterogeneous edge devices," *IEEE/ACM Transactions on Networking*, 2021.

[4] J. Mohan, A. Phanishayee, A. Raniwala, and V. Chidambaram, "Analyzing and mitigating data stalls in DNN training," *PVLDB*, vol. 14, 2021.

[5] Z. Wang, Z. Wang, C. Liu, and Y. Hu, "Understanding and tackling the hidden memory latency for edge-based heterogeneous platform," in *USENIX HotEdge*, 2020.

[6] H. A. Abdelhafez, H. Halawa, K. Pattabiraman, and M. Ripeanu, "Snowflakes at the edge: A study of variability among NVIDIA Jetson AGX Xavier boards," in *ACM EdgeSys Workshop*, 2021.